UNIVERSITY OF CALIFORNIA SAN DIEGO

Data Conditioning and Data Assimilation for Wildfire and Prescribed Fire Modeling

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy

in

Engineering Sciences (Mechanical Engineering)

by

Li Tan

2023

DEDICATION

To my beloved family.

EPIGRAPH

Somewhere, something incredible is waiting to be known.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge every person who inspired, encouraged, and supported me during my school life at UC San Diego. First of all, I would like to express my deepest appreciation to my advisor, Dr. Raymond de Callafon, for his invaluable guidance and support. Raymond is insightful, enthusiastic and assiduous. Meanwhile, he is humorous and considerate. Besides professional advice, he also helps me a lot in my daily life. I am extremely grateful that he provided me with an valuable opportunity to study for my doctorate.

I would also like to extend my deepest gratitude to my doctoral committee of Dr. Kramer, Dr. Hidalgo-Gonzalez, Dr. Wang, Dr. Kriegman and Dr. Altintas for their constructive comments that greatly improved the quality of this dissertation. I am particular thankful to Dr. Kriegman for his remarks on the classic edge detection techniques.

I am deeply indebted to Dr. Altintas for supporting my research. Thanks should also go to Mai Nguyen, Daniel Crawl, Jessica Block, Matt Snider, Kevin Hiers, Rukmini Ravi, Tolga Çağlar and all other people at San Diego Supercomputer Center for their meaningful thoughts and beneficial opinions. It has been a wonderful and unforgettable experience to work with all of you.

I had the pleasure of working with my labmates Yangsheng Hu, Yunfeng Jiang and Amir Valibeygi. They are all talented, self-motivated and friendly. Special thanks to Yangsheng Hu for his inspiration and helpfulness. Additional thanks to Tianshi Feng, Han Zhao, Yuzhe Qin, Jie Feng, Aobo Yang, Chenghai Li, Cuncheng Zhu, Bingran Wang, Kuan Feng, Muhan Zhao, Jiajie Shi, Pengcheng Cao and other dear friends, for making my life at UC San Diego more pleasant. My best regards to all of you.

At last, I cannot begin to express my thanks to my family for their love, understanding, patience and support. Thanks for being my rock and I feel lucky and grateful to have you all in my life.

Chapter 2, in full, is a reprint of the material as it appears in "Estimation of wildfire wind conditions via perimeter and surface area optimization", Tan, L., de Callafon, R. A., Block, J.,

Crawl, D., Çağlar, T., & Altıntaş, I. (2022). Journal of Computational Science, 61, 101633. The dissertation author was the primary investigator and author of this paper.

Chapter 3, in full, is a reprint of the material as it appears in "Characterizing Wildfire Perimeter Polygons from QUIC-Fire", Tan, L., de Callafon, R. A., & Altıntaş, I. (2022, June). In Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part I (pp. 611-622). Cham: Springer International Publishing. The dissertation author was the primary investigator and author of this paper.

Chapter 4, in full, has been accepted for publication of the material as it may appear in "Wildfire Perimeter Detection via Iterative Trimming Method", Tan, L., Hu, Y., Tan, S., de Callafon, R. A., & Altıntaş, I. (2023, July). In Computational Science–ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 5, in full, has been accepted for publication of the material as it may appear in "Ensemble Based Learning for Automated Safety Labeling of Prescribed Fires", Tan, L., de Callafon, R. A., Nguyen, M., & Altıntaş, I. (2023, July). In Computational Science–ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 6, in full, is currently being prepared for submission for publication of the material. Tan, L., de Callafon, R. A., Nguyen, M., & Altıntaş, I. The dissertation author was the primary investigator and author of this material.

VITA

2016	Bachelor of Science in Engineering, Shandong University
2018	Master of Science, Virginia Polytechnic Institute and State University
2023	Doctor of Philosophy, University of California San Diego

PUBLICATIONS

Tan, L., Hu, Y., Tan, S., de Callafon, R. A., & Altıntaş, I. (2023, July). Wildfire Perimeter Detection via Iterative Trimming Method. In Computational Science–ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023, accepted.

Tan, L., de Callafon, R. A., Nguyen, M., & Altıntaş, I. (2023, July). Ensemble Based Learning for Automated Safety Labeling of Prescribed Fires. In Computational Science–ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023, accepted.

Tan, L., de Callafon, R. A., Block, J., Crawl, D., Çağlar, T., & Altıntaş, I. (2022). Estimation of wildfire wind conditions via perimeter and surface area optimization. Journal of Computational Science, 61, 101633.

Tan, L., de Callafon, R. A., & Altıntaş, I. (2022, June). Characterizing Wildfire Perimeter Polygons from QUIC-Fire. In Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part I (pp. 611-622). Cham: Springer International Publishing.

Hu, Y., Tan, L., & de Callafon, R. A. (2022). Noniterative tensor network-based algorithm for Volterra system identification. International Journal of Robust and Nonlinear Control, 32(9), 5637-5651.

Tan, L., de Callafon, R. A., Block, J., Crawl, D., & Altıntaş, I. (2021, June). Improving wildfire simulations by estimation of wildfire wind conditions from fire perimeter measurements. In Computational Science–ICCS 2021: 21st International Conference, Krakow, Poland, June 16–18, 2021, Proceedings, Part V (pp. 231-244). Cham: Springer International Publishing.

Subramanian, A., Tan, L., de Callafon, R. A., Crawl, D., & Altintas, I. (2020). Recursive updates of wildfire perimeters using barrier points and ensemble Kalman filtering. In Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part VI 20 (pp. 225-236). Springer International Publishing.

Hu, Y., Tan, L., & de Callafon, R. A. (2019, December). Persistent excitation condition for MIMO Volterra system identification with Gaussian distributed input signals. In 2019 IEEE 58th

Conference on Decision and Control (CDC) (pp. 1752-1757). IEEE.

ABSTRACT OF THE DISSERTATION

Data Conditioning and Data Assimilation for Wildfire and Prescribed Fire Modeling

by

Li Tan

Doctor of Philosophy in Engineering Sciences (Mechanical Engineering)

University of California San Diego, 2023

Professor Raymond A. de Callafon, Chair

Wildfire is an unplanned fire that can happen at any time in a natural area with combustible materials. Normally, there are two main reasons for the occurrence of a wildfire. One is due to human activity and the other one is owing to natural phenomenon, such as lighting, volcanic eruption, or even falling meteors. Although some naturally occurring wildfires may be beneficial for the ecological balance, most of the wildfires are destructive. They can lead to air pollution and may bring death and destruction. As the climate gets warmer and drier, the occurrence of wildfire is more frequent. Consequently, the estimation and prediction of the spread of wildfire is significant. Prescribed fires are the planned fires that burn under specified conditions to achieve specific objectives. Compared to wildfires, prescribed fires are implemented and controlled in a safer way to balance ecosystems. Meanwhile, prescribed fire can also be used as a tool to reduce fuel build-up and avoid the occurrence of wildfires. Hence, how to design a safe and effective burn plan for the prescribed fire is a good topic for research. This dissertation proposes multiple algorithms for estimation and prediction of wildfire spread, detection of the wildfire perimeter, and safety evaluation of the prescribed fire. First, this dissertation shows how to improve the prediction capability of a fire model by estimating the wind conditions. Two errors, an uncertainty-weighted least-squares error and an uncertainty-weighted surface area error, are established and computed between the predicted and measured fire perimeters, and an optimization is applied to find the optimal wind conditions based on iterative refined gridding of the wind conditions. To better characterize the wildfire for the estimation and prediction of the fire spread, three approaches are introduced to detect the wildfire perimeter in different situations. The first two approaches, quadriculation algorithm and iterative minimum distance algorithm, are used to establish a closed polygon of the wildfire perimeter for a well-defined fire image, and the third one, iterative trimming method, is based on the Delaunay triangulation and designed for the situation when the pixels with high infrared values in a thermal infrared image of the wildfire are disconnected or sparse. In addition to managing wildfire directly, the prescribed fire can be utilized to reduce and prevent the wildfire. An algorithm of automatically labeling the safety of a prescribed fire is proposed in this dissertation to avoid a labor-intensive process of manual labeling. All the proposed algorithms are illustrated on real data of a wildfire or simulations from modern fire simulation tools.

Chapter 1

Introduction

1.1 Wildfire and Prescribed Fire

A wildfire is an unplanned and uncontrolled fire. Fire historically plays an important role in the earth system [1]. Although wildfire can have a negative effect on the quality of the air and drinking water, it is also a natural part in many fire-dependent ecosystems, such as forest and grassland ecosystems. Wildfire can help an ecosystem to regenerate and create new habitats for wildlife. However, with the increase of the temperature, wildfire becomes more and more frequent and intense [2]. A long-time large-scale wildfire can even hinder the recovery of the ecosystem. As a consequence, cutting-edge research of the wildfire science is necessary and can be adopted by fire managers and scientists to understand how to take advantage of the beneficial aspect of the wildfire and avoid the environmental harm from the wildfire. Currently, the research portfolio of the fire science summarized by U.S Forest Service includes five main aspects: fire behavior, fuels management, social fire science and risk management, smoke and air quality, and the safe and effective fire response.

The dynamic of fire progression in a landscape with a specific spatial scale is studied to explore the fire behavior. It is a fundamental physical fire science for the fire model and is relied on to predict the fire spread. A mathematical model was introduced by Rothermel to predict the progression of the wildfire [3]. At the beginning, the fire spread is modeled empirically by some easily measured variables, such as fuels and weather conditions, and few physical mechanisms are included to improve the computation speed. Specifically, the process of combustion and heat transfer of the wildfires are simplified and the interaction between the wildfire and the surrounding atmosphere is not considered. As a result, the traditional wildfire simulation tool cannot fully capture the process of the fire growth, and the prediction of the wildfire spread from the traditional approach needs to be improved in many situations. To this end, data assimilation techniques are applied to adjust the model prediction by using the measurement of the wildfire [4–7]. Among these simulation tools for wildfire, FARSITE [8] is widely used to simulate the propagation of wildfires spatially and temporally under burn conditions that mainly consist of wind conditions, fuel moisture and landscape information. With the development of fire methodologies, physical system is cautiously established and more complete mechanism of the combustion, heat transfer and fluid dynamics are adopted in the wildfire model [9–12]. In addition, the interaction between the fire and the weather conditions surrounding the fire is also included. QUIC-Fire [13], a fast-running simulation tool, is designed to simulate the progression of fuel consumption for a specific landscape in three dimensions. During the simulation, the dynamic coupling between the fire and the surrounding atmosphere is also approximated. QUIC-Fire has a resolution of one meter and can also model the interactions between multiple fires.

Fuels management mainly consists of prescribed fire and mechanical cutting. Fuels management can result in many beneficial impacts including protecting high-value areas, restoring ecological processes, avoiding hazardous wildfire and species invasion, and so forth. Little scientific attention is paid to prescribed fire [14]. The importance of the prescribed fire is also highlighted in [15]. Compared to an unplanned wildfire, prescribed fire is relatively low-severity and can be designed to minimize the adverse effect of an uncontrollable fire. Instead of being regarded as a risk, the prescribed fire can serve as a good tool to avoid catastrophic wildfire by reducing the vegetative fuel and can be used for forest management by maintaining the system resilience. Due to the fact that an urgent response is required for a wildfire, while a detailed burn plan is afforded for a prescribed fire, the planning horizon is different for them. Therefore, studies of wildfire may not be appropriate for the prescribed fire, and more research should be conducted to learn the behavior and influence of a prescribed fire.

The interconnection between fire and our community and society is intricate and worthy of study. For a wildfire, complexity of fire management system, decision making, and health and safety improvements are emphasized in the risk management. Many studies have proven that the socioecological system (SES) approach is a good way to address this interdisciplinary issue, and a review of the literature on the application of SES frameworks can be found for wildfire risk management [16].

Wildfire is a major cause of air pollution and can be harmful to public health. Specifically, the smoke produced by a wildfire or a prescribed fire lowers the air quality and can even lead to violations of state and federal air quality standards. Moreover, firefighters are likely to be exposed to dense smoke in the process of putting out a wildfire. In the research of smoke management, emission characterization, emission inventory, and smoke impact are three key things. Many tools have been developed to represent the essential processes of smoke transportation and predict the effects of the smoke [17].

In addition to the exposure of the smoke, wildfire can also give rise to many other risks threatening the safety of both community and firefighters. These risks should be first well explored before the establishment of a safe and effective response to a wildfire. Figuring out why a fire can rapidly grows and finding out possible firefighters entrapments and fatalities are necessary to prevent the potential injury and death. Many researches have been done in developing a safe and efficient response to a wildfire [18–21].

This dissertation is focused on improving the prediction accuracy of the fire spread and evaluating the safety of a prescribed fire. No matter for a wildfire or a prescribed fire, safety is the most important thing. How to manage a wildfire by accurately predicting and estimating the progression of the fire and how to design a burn plan for a prescribed fire remain to be good research topics. In addition to concentrating on the fundamental process of the fire spread, how to apply the techniques in other fields, such as image segmentation methods, on the fire image to detect the wildfire perimeter is also meaningful for predicting the spread of a fire.

1.2 Motivations

As mentioned before, prediction of wildfire spread is important for the management of a wildfire. Due to the fact that the processes of combustion and heat transfer are simplified and the fire and atmosphere are not coupled, traditional fire models cannot capture the complete progression of the wildfire. To solve this problem, data assimilation technique, such as ensemble Kalman filter, is applied to improve the prediction accuracy. Measurement of the wildfire and the simulation from the fire model are both taken into consideration to find an optimal estimation of the wildfire spread. Although the prediction accuracy of the fire spread increases a lot, it can be further enhanced by improving the accuracy of the measurement and simulation of the wildfire.

FARSITE [8] is widely used for the forward simulation of the wildfire as a function of the landscape and weather information includes the wind speed, wind direction, fuel moisture and terrain. Among them, wind speed and wind direction play an important role in the fire spread [22]. However, the information of wind conditions can only be obtained from sparsely placed weather stations and are often empirically estimated. In reality, the wind conditions surrounding the wildfire are hardly estimated from an empirical formula due to the limited number of weather stations and the coupled effects between the atmosphere and the wildfire. As a result, significant and compounding errors can occur in the data of wind conditions, which will lead to an unreliable result simulated by the fire model. Therefore, how to correctly estimate the wind conditions becomes a crucial problem before any data assimilation technique is applied.

With the development of science in the field of wildfire, more and more advanced fire simulation models have been established. QUIC-Fire [13] is a modern simulation tool designed to simulate the progression of the fuel consumption. Although people can realize the burn area of a fire from the plot of the fuel consumption, it is desirable to create a closed polygon for the fire perimeter automatically. The closed polygon is a numerical representation of a fire perimeter and can be used for various computations. For example, the closed polygon of a wildfire perimeter is an important component for the procedure of previously mentioned data assimilation. Despite the fact that unordered vertices of the polygon can be detected by traditional edge detection methods when the fire image is clear and complete, how to reorder those vertices to establish a closed polygon is not trivial. Furthermore, since embers happen frequently during the spread of a fire, multiple burn areas need to be detected and closed polygons should be created for each burn area. In addition to vertices, a closed polygon can also be established by joining the internal adjacent polygons. Hence, how to find all sub-polygons inside a fire perimeter and join them is another way of creating a closed polygon for the fire perimeter.

As sensor technology advances in the recent years, more and more monitoring technique can be used to measure the spread of a wildfire. For example, wildfires can be monitored by characterizing ground temperature via MODIS data [23], satellite heat images [24] or thermal infrared imaging (TIR) on aerial flight systems [25]. For TIR or heat images, temperature information or strength of the infrared band are represented by image pixels. A wildfire perimeter is then expected to be extracted from a TIR image of a wildfire. Currently, manual delineation is still one of the main ways to obtain the wildfire perimeter. In the meantime, due to limited resolution of the image and possibly partial activity of a wildfire, the burn area in a TIR image can be disconnected or even sparse, and automatic characterization of wildfire perimeter is still challenging. Although many traditional image segmentation methods are applied to a well-established fire image (with clear and connected burn area) and decent results can be achieved, they have not demonstrated their abilities to detect a wildfire perimeter from a TIR image with disconnected or even sparse burn area. Machine learning or deep learning might also be a good tool to learn the characteristic of a wildfire in the future. However, the arbitrary shape of a wildfire and the fact that a large amount of training data is required by artificial intelligence techniques are still great restrictions. Therefore, how to automatically detect the wildfire perimeter in a limited number of TIR images with sparse data remains a challenge.

As introduced before, prescribed fire has many positive effects, such as balancing an ecosystem and avoiding severe wildfire. However, few research has been done for the prescribed fire, and the difference between the wildfire and the prescribed fire leads to a result that the studies of wildfire cannot be applied straightforwardly to a prescribed fire. As a result, exploration in the field of the prescribed fire is a good research direction. Safety is certainly one of the most important problems that needs to be explored since a good burn plan for a prescribed fire must ensure that the planned prescribed fire achieves the goal of management in a safe manner. QUIC-Fire is a fire simulation tool that is developed to provide dynamic fuel consumption as a function of burn conditions including the wind conditions, fuel moisture, ignition patterns and so forth. To determine the proper burn conditions, plenty of simulations are required to be done, and fire domain experts need to decide (label) whether a simulated prescribed fire is safe and the corresponding burn conditions are acceptable. Manual labeling is a labor-intensive and time-consuming process, and people are more likely to make a mistake after long hours of work. As a consequence, an algorithm of automatically labeling the safety of a simulated prescribed fire is meaningful.

1.3 Problem Formulation

On the basis of the previously mentioned challenges in fire science, this dissertation is an attempt to deal with the prediction accuracy of fire progressions, the detection of fire perimeters, and the fire safety evaluation of prescribed fires. Specifically, this dissertation investigates the following problems:

1. Improve the prediction accuracy of a wildfire spread by using the measurements of the wildfire perimeters to estimate and correct the prevailing wind speed and wind direction for the simulation. This problem is studied in Chapter 2.

2. Detect fire perimeters from well-established fire images and discontinuous fire images with sparsely located high-value pixels. This problem is discussed in Chapters 3 and 4.

3. Determine the safety of a simulated prescribed fire by evaluating the metrics of the slop-over generated in each simulations. This problem is covered in Chapters 5 and 6.

1.4 Organization and Contributions

In Chapter 2, it is shown that the prediction accuracy of fire simulation tool FARSITE can be improved by optimizing the inputs of wind speed and wind direction. The optimization of the prevailing wind speed and wind direction can be achieved by comparing the simulated wildfire perimeters from FARSITE and measured wildfire perimeters. Uncertainties are established at each vertex of measured wildfire perimeters to model the noisy observation of the wildfire perimeters. Two errors from two perspectives are proposed to describe the difference between the simulated wildfire perimeters and measured wildfire perimeters. The first one is an uncertainty-weighted least-squares error that measures the difference by considering the locations of the vertices of the wildfire perimeter, and the second one is an uncertainty-weighted surface area error that determines the difference by computing the surface area of the union minus the intersection of the simulated and measured wildfire perimeters. For the uncertainty-weighted least-squares error, the vertex number of the simulated wildfire perimeters and the measured wildfire perimeters are matched via interpolation. For the uncertainty-weighted surface area error, the relation between the surface area error and the uncertainties at each vertex of the wildfire perimeters are set up. With these two errors, the optimization is done by an iteratively refined grid search in wind speed and wind direction. The grid search method is robust to the occasional erroneous results produced by FARSITE and can be executed in parallel in order to reduce the computation time.

In Chapter 3, two algorithms, iterative minimum distance algorithm (IMDA) and quadriculation algorithm (QA) are compared to quantitatively characterize the closed polygons of wildfire perimeters. For IMDA, some classical image segmentation methods should be applied to detect the unordered boundary points of the wildfire perimeter, and a threshold value should be established to determine whether two vertices are adjacent or closely located. An initial closed polygon by connecting the closest points one by one in the set of boundary points is first created. Then the created polygon is iteratively modified when the distance between two neighboring points is larger than the established threshold value until the distance between every pair of adjacent points is smaller than the threshold value. Multiple polygons should be created if one cluster of unordered boundary points is far away from the others. From a different perspective, QA creates the polygon in two steps. At first, QA recursively divides the raster image into indivisible squares or rectangles, where all the pixels in an indivisible square or rectangle have the same value. Then the adjacent squares or rectangles with same pixel value are merged, and the closed polygon of the wildfire perimeter can be obtained by combining the closed polygon of the adjacent squares or rectangles with the same pixel value that represents the burn area. Both QA and IMDA are applied to the simulation data of the fuel consumption produced by QUIC-Fire and the performances of the two algorithms are compared.

In Chapter 4, an iterative trimming method is introduced to create a closed polygon of the wildfire perimeter for a thermal infrared (TIR) image with disconnected or even sparse high-value pixels. For a TIR image, all the (active) pixels that represent the active wildfire are picked out and Delaunay triangulation are applied based on these active pixels to connect the disconnected burn area. Then a convex hull that covers the wildfire perimeter is established by joining the adjacent triangles created by Delaunay triangulation. To obtain the closed polygon of the wildfire perimeter that hides in the convex hull, two steps of iterative trimming, rough trimming and fine trimming, are introduced to remove the redundant triangles that are created by connecting the vertices of the wildfire perimeters or by active pixels caused by spot fires. During the process of the rough trimming, the abnormally large triangles caused by the vertices of the wildfire perimeters can be detected by the longest side of the triangle, and the active pixels caused by the spot fires can be detected by the relative burn area around the active pixels in a chosen domain. After the rough trimming, all the active pixels that represents the spot fires and parts of the abnormally large triangles are deleted. Fine trimming is utilized to further trim the closed polygon created based on the vertices determined in the rough trimming. During the process of the fine trimming, no more vertices are removed, and only the triangles connecting to the vertices can be cut out so that no hole is created inside the closed polygon of the wildfire perimeter. The redundant triangles connecting the vertices of the closed polygon are identified by considering the combined effect of the longest side of triangle and the relative burn area around the vertex. Larger relative burn area provides more detailed information of the wildfire, and greater restriction is applied to the longest side of the corresponding triangle. On the contrary, smaller relative burn area indicates a higher possibility of missing information related to the wildfire. In this situation, more cautions are put into removing a triangle. The closed polygon established by the iterative trimming method is compared to the results obtained by the classical image segmentation methods, such as canny edge detector, graph-cut method, and level set method.

In Chapter 5, an automatic labeling algorithm is proposed to label the safety of a simulated prescribed fire. Inspired by the fire domain experts, three metrics including the number of the slop-overs, the total surface area of the slop-overs, and the distance between each slop-over are established to measure the safety of a simulated prescribed fire. To assist in the measurement, multiple parameters, such as maximum allowable total surface area of the slop-overs, marginally allowable total surface area of the slop-overs, maximum allowable distance between each slop-over, marginally allowable distance between each slop-over, maximum allowable number of the slop-overs and so forth are set up. To optimize these parameters, an objective function is designed with the purpose of increasing the match number between the labels created by automatic labeling algorithm and the manual labels created by fire domain experts, and genetic algorithm is adopted for the optimization. 48 runs out of 900 simulations (ensembles) with two sets of manual labels from two fire domain experts are used as the training data, and the other 852 runs out of 900 ensembles with one set of manual labels created together by the two fire domain experts are used to validate the accuracy of the automatic labeling algorithm. In addition to labeling the safety of a simulated prescribed fire automatically, automatic labeling algorithm also has the ability to provide the explanation why a prescribed fire is considered to be unsafe or marginal.

In Chapter 6, another logic model is established to automatically create the fire safety label of a prescribed fire. Compared to the previous automatic labeling algorithm proposed in Chapter 5, the main development is the exploitation of the fire growth rate. Instead of the sophisticated logic rule created in the previous algorithm, the new logic model is more general and comprehensive by including the velocity of the fire growth. The fire growth at each time step is filtered to remove the high-frequency noise term, and the maximum velocity is picked out differently depending on whether the simulated prescribed fire escapes the predetermined allowable boundary to capture the worst situation of the fire spread. The same 900 simulated prescribed fires in Yosemite, CA region, are used for optimizing the parameters created in the logic model and validating the performance of the new automatic labeling algorithm. A similar accuracy of matching is achieved between the automatic labels and manual labels due to the fact that wind speed and surface moisture can reflect the fire growth to some extent. To further analyze the inconsistency of the manual labels, the minimum and maximum value of each metric used for the logic model are computed for the manually labeled safe, marginal, and unsafe fire on the basis of the 48 ensembles in the training data set and 852 ensembles in the validation data set.

Chapter 2

Estimation of Wildfire Wind Conditions via Perimeter and Surface Area Optimization

This chapter shows that the prediction capability of wildfire progression can be improved by estimation of a single prevailing wind vector parametrized by a wind speed and a wind direction to drive a wildfire simulation created by FARSITE. Estimations of these single wind vectors are achieved in this work by a gradient-free optimization via a grid search that compares wildfire model simulations with measured wildfire perimeters, where noisy observations are modelled as uncertainties on the locations of the vertices of the measured wildfire perimeters. Two optimizations are established to acquire the optimal wind speed and wind direction. To formulate a perimeter optimization, an uncertainty-weighted least-squares error is computed between the vertices of simulated and measured wildfire perimeters. The challenge in this approach is to match the number of vertices on the simulated and measured wildfire perimeters via interpolation of perimeter points and their uncertainties. For a surface area optimization, an uncertainty-weighted surface area error is introduced to capture the surface of the union minus the intersection of the predicted and measured wildfire perimeters. The challenge in this approach is to formulate a surface area error, weighted by the uncertainties on the locations of the vertices of the measured wildfire perimeter. The optimization in this chapter is based on iterative refinement of a grid of the single wind vector and provides robustness to intermittent erroneous results produced by FARSITE, while allowing parallel execution of wildfire model calculations. This chapter is an extension of the work in [26]. Results on single wind vector estimation are illustrated on two historical wildfire events: the 2019 Maria Fire that burned south of the community of Santa Paula in the area of Somis, CA, and the 2019 Cave Fire that started in the Santa Ynez Mountains of Santa Barbara County.

2.1 Introduction

With the increased and inevitable occurrence of wildfires, more accurate and responsive prediction of the wildfire propagation is important for resource allocation in fire fighting efforts. The wildfire growth modeling software FARSITE is widely used by the U.S Forest Service to simulate the propagation of wildfires [8], and is characterized by the ability to estimate the wildfire propagation under heterogeneous conditions of terrain, fuels and weather. Crucial source of information in the modeling of fire progression is a single wind vector characterized by average wind speed and wind direction that determine the overall direction and rate of spread of the wildfire. This chapter is an extension of the work in [26] by adding the ability to formulate an uncertainty-weighted wildfire surface coverage error to estimate the single wind vector of wind speed and wind direction, which is explained in Section 2.3.2. The numerical results are summarized in Section 2.5.

The prediction of the growth of wildfires has received a considerable amount of attention in the literature. Rothermel introduced the mathematical model for predicting fire spread [3], and experiments have been conducted to analyse the influence of fuel and weather on the spread of wildfires [22]. Further steps in the study of the wildfire behavior were achieved by adjusting model prediction using real-time data via data assimilation techniques [4–6]. Data assimilation by combining FARSITE and an ensemble Kalman filter has been done in earlier work [7,27–29] demonstrating an improvement in accuracy of wildfire prediction. The availability of unmanned aerial vehicles to better monitor large-scale wildfire [30,31] has further enhanced the capabilities of data-driven wildfire modeling.

As mentioned in [22], among the numerous factors that can affect the spread of the wildfire, wind speed and wind direction play the critical roles. Unfortunately, wind conditions are available only from sparsely placed weather stations. Detailed studies are available on learning the (non-linear) relationship between the properties of the fuel and the wildfire progression [32–34], but often only limited information on wind speed and wind direction can be used. This means that the quality of the prediction is extremely dependent on the quality of an empirical estimate of the wind conditions obtained from geometrically spaced weather station. In reality, information of the actual wind conditions at the boundary of the wildfire is unavailable due to limited number of weather stations and the turbulent atmosphere caused by wildfire. As a result, significant and compounding errors can occur in the prediction of the wildfire propagation. A first step is to estimate the best initial wind conditions before any data assimilation procedure. In this situation, the error caused by an erroneous measurement of the wind conditions can be reduced, and the accuracy of the prediction by data assimilation techniques can be greatly improved.

A gradient-free optimization via a grid search is used in this work to provide an estimate of the single wind vector of wind speed and wind direction fed to FARSITE with the objective to improve the prediction of wildfire progression. The gradient-free optimization via a iterative grid search refines a grid of wind speed and wind direction to select the best single wind vector based on a loss function that compares wildfire model simulations with noisy observations of the wildfire perimeters. Since each grid point provides an independent wildfire simulation, the computations can be executed in parallel and also provides robustness to possible erroneous perimeter produced by FARSITE under certain single wind vector. To formulate the loss function, it is first shown that noisy observations can be modelled as uncertainties on the locations of the vertices of a measured wildfire perimeter. Secondly, it is shown that a uncertainty-weighted error can be computed between the vertices of a simulated wildfire and a measured wildfire perimeter.

In this chapter, two different uncertainty-weighted errors are formulated for the estimation of the single wind vector: a perimeter and surface area based. For the perimeter optimization, a skew compensated and uncertainty weighted least-squares error is computed between the vertices of a simulated and measured wildfire perimeter. To be able to compute this perimeter error, it is shown that a linear interpolation of the perimeter is used to guarantee that the skew compensated weighted least-squares error can always be computed. Furthermore, compared to an ordinary weighted least-squares error, the weighting in the skew compensated weighted least-squares computation is adjusted to account for unevenly distributed polygons to allow an evenly distributed weighting of the complete wildfire perimeter. The surface area optimization captures the wildfire surface area error defined by the union minus the intersection of a simulated and measured wildfire perimeter. By using the fact that the surface area of a closed polygon can be calculated as the signed sum of triangular sub-polygons [35, 36], it is shown how to compute an uncertainty-weighted surface area error. The weighting is again determined by the uncertainties on the locations of the vertices of the measured wildfire perimeter and the computational process of the weighted surface area error is simple and fast.

The chapter is organized as follows. Section 2.2 presents the model of the polygon data along with the uncertainties on the vertices of a wildfire perimeter. Following the uncertainty characterization, Section 2.3 presents the computations of the skew compensated uncertainty-weighted least-squares error and the uncertainty-weighted surface area error. Section 2.4 outlines the parallel gradient-free optimization via a grid search based on refining a grid of wind speed and wind direction to estimate the best single wind vector. Section 2.5 shows the numerical results for the estimation of the single wind vector for two use cases of wildfires in California: the 2019 Maria Fire that burned south of the community of Santa Paula and the 2019 Cave Fire that started in the Santa Ynez Mountains of Santa Barbara County. Conclusions are summarized in Section 2.6.

2.2 Wildfire Perimeter and Uncertainty

A wildfire may cover multiple disjoint burned areas. For simplicity of the analysis presented in this chapter, the notion of wildfire progression is characterized by a wildfire perimeter that is considered to be a single closed polygon. The analysis presented here can be applied to each of the closed-polygons in case a wildfire does cover multiple disjoint burned areas. The single closed polygon describing the wildfire perimeter is an ordered sequence of N vertices and N piece-wise linear line segments. The vertices of the approximated polygon are located by the Eastern and Northern coordinate pairs $(e(k), n(k))$, $k = 1, 2, \ldots, N$.

2.2.1 Uncertainty Characterization

Measurements of the wildfire perimeters can be a combined data collection effort. The resolutions and spacing of the measured vertices are determined by data from satellite imagery, aerial surveillance or manually mapped observations. [37]. Therefore, it is important to consider the two-dimensional (2D) uncertainty for each vertex of the closed polygon that describes the measured wildfire perimeter. The general description of the 2D uncertainty on a vertex $(e(k), n(k))$ is a rotated ellipse, where the semi-major axis $a(k)$, semi-minor axis $b(k)$, and the rotation angle $\alpha(k)$ collectively reflect the variance in the horizontal direction and vertical direction. Such detailed information may not be available and therefore the uncertainty on a vertex $(e(k), n(k))$ is expressed by a circle around each vertex with a radius $r(k)$, where the value of $r(k)$ is proportional to the uncertainty of the vertex on the polygon.

However, it is very likely that a measured perimeter comes with no additional uncertainty characterization. In that case, the assumption is made that the uncertainty on each vertex is proportional to the (smallest) distance to the neighboring vertex on the polygon. The reason for that is the vertices are more likely to have large uncertainties for sporadic measurements with a large distance between the vertices. Formally this uncertainty is described by

$$
\begin{aligned}
r(k) &= \max(\min(l(k), l(k-1)), r_{min}) \\
l(k) &= \sqrt{(e(k+1)-e(k))^2 + (n(k+1)-n(k))^2}
\end{aligned}
\tag{2.1}
$$

for $k = 1, 2, \ldots, N$, where $r(k)$ is the assumed uncertainty, $l(k)$ is the distance between neighboring vertices $(e(k+1), n(k+1))$, $(e(k), n(k))$, and r_{min} is a user-defined minimum value of uncertainty radius. The value of r_{min} is used to avoid the condition in which two adjacent vertices are extremely close to each other, and can be determined by the accuracy of measuring method used to acquire the polygon of the wildfire perimeter. An illustration of the uncertainty assignment for a measured wildfire perimeter is given in Fig. 2.1.

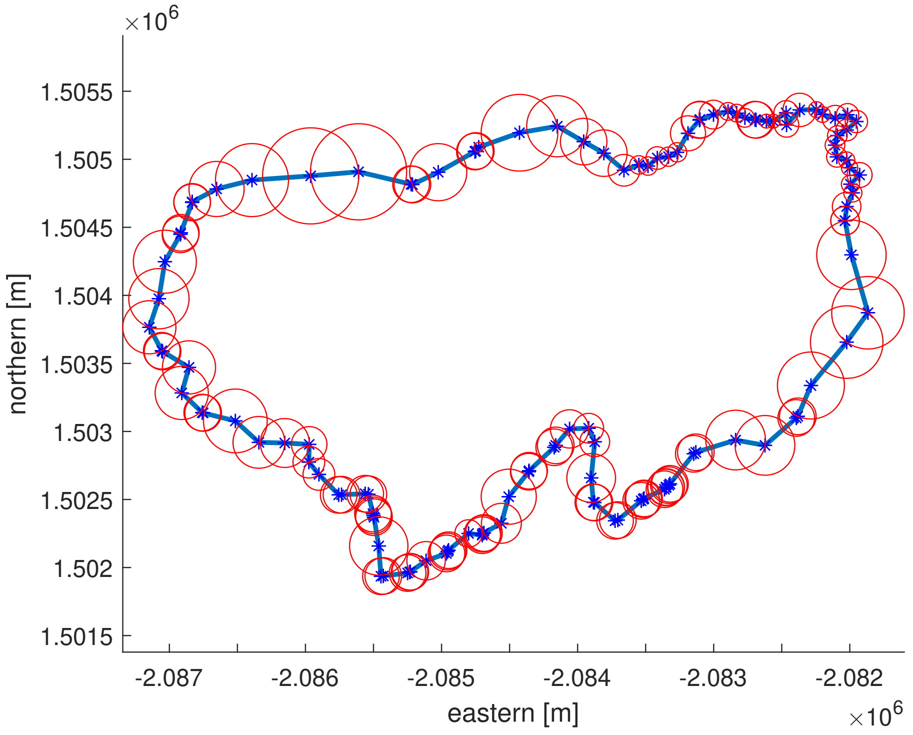


Figure 2.1. Assignment of uncertainty radii $r(k)$ (red circles) on a measured wildfire perimeter with vertices $(e(k), n(k))$ (blue stars) and the resulting closed polygon (blue lines).

2.2.2 Perimeter Interpolation

With the spread of a wildfire, the corresponding closed polygon describing the measured wildfire perimeter commonly becomes larger and the number N_m of vertices of the measured closed polygon $(e_m(k_m), n_m(k_m))$, $k_m = 1, 2, \ldots N_m$ increases accordingly. Similarly, the number of vertices N_s on a simulated wildfire perimeter $(e_s(k_s), n_s(k_s))$, $k_s = 1, 2, \ldots N_s$ obtained with fire modeling software such as FARSITE will also increase, but in general $N_m \neq N_s$. The resolution of the simulated vertices is determined by the fire modeling software FARSITE and typically in the order of 30 meters. Next to difference in number of vertices, the ordering of the vertices $(e_m(k_m), n_m(k_m))$, $k_m = 1, 2, \ldots N_m$ of the measured fire perimeter and $(e_s(k_s), n_s(k_s))$, $k_s = 1, 2, \ldots N_s$ are not the same and a direct comparison between a pair of vertices $(e_m(k_m), n_m(k_m))$ and $(e_s(k_s), n_s(k_s))$ would lead to erroneous results.

A direct comparison of a measured vertex $(e_m(k_m), n_m(k_m))$ and a simulated vertex $(e_s(k_s), n_s(k_s))$ is especially important if a (weighted) least-squares error based on vertices needs to be formulated. To anticipate the notion of an uncertainty weighted least-squares error, it is shown how perform a interpolation of the wildfire perimeter to create $N_m = N_s$ and therefore an equal number of N_m of vertices of the measured closed polygon and N_s of vertices of the simulated closed polygon. The solution to this problem is to first interpolate one of the fire perimeters to the same or higher number $N = \max(N_m, N_s)$ of vertices of the other fire perimeter. Subsequently, when comparing pairs $(e_m(k_m), n_m(k_m))$ and $(e_s(k_s), n_s(k_s))$, the starting vertex at $k_m = 1$ or $k_s = 1$ of one of the fire perimeters will be re-ordered to obtain the smallest weighted least-squares error between the polygons.

For simplicity of notation, k is used to represent both k_s and k_m after the interpolation because the simulated fire perimeter and the measured fire perimeter have the same number of vertices. In this chapter, interpolation of the fire perimeter is done with standard 2D linear interpolation, where interpolated vertices are introduced on the straight lines connecting the original vertices of the closed polygon, and the procedure of linear interpolation is summarized in Algorithm 1.

Algorithm 1. Linear interpolation of wildfire polygon

```
Input: vertices of the original approximated polygon
Output: Newly constructed vertices of the interpolated polygon
1: Calculate the length of each side of the polygon.
2: Calculate the cumulative side length from the starting point.
3: Find locations with equally distributed length along the side of polygon from the starting
   point.
4: Construct new polygon vertices
```

Similarly, uncertainties of the original vertices can also be interpolated with respect to the cumulative side length from the starting point. Due to the fact that the interpolation is related to the distance from the starting point, it is easy to verify that interpolation from different starting points will lead to different results. This will be considered in the subsequent section when the weighted least squares are calculated. Linear interpolation may lead to a tiny change of the shape of the original wildfire polygon that is negligible compared to the huge burned area of the wildfire polygon. Therefore, the change of the wildfire polygon caused by the linear interpolation is not considered in this chapter.

2.3 Wildfire Error Quantification

2.3.1 Weighted Least-Squares Error

With an interpolated (and properly ordered) closed polygons of the simulated fire perimeter $(e_s(k), n_s(k))$, and the measured fire perimeter $(e_m(k), n_m(k))$ with an uncertainty $r(k)$ on each vertex $k = 1, 2, \dots, N$, a weighted least-squares error

$$\frac{1}{N}\sum_{k=1}^{N} w(k)^2 \left[\left(e_s(k) - e_m(k)\right)^2 + \left(n_s(k) - n_m(k)\right)^2\right], \quad w(k) = \frac{1}{r(k)} \tag{2.2}$$

can be used to define the distance between the fire perimeters. The interpolated weighting $w(k) = 1/r(k)$ ensures measurements with a large uncertainty $r(k)$ are weighted less in the error characterization. However, even with uncertainty radii defined by (2.1) with a minimum value r_{min}, the weighted least-squares error in (2.2) will be skewed and emphasizes parts of the closed-loop polygon where vertices are closely clustered and have only small distances with respect to each other, as also illustrated in Fig. 2.1. The reasons are clear:

- Small uncertainty radii $r(k)$ due to (2.1) will result in a larger weighting $w(k) = 1/r(k)$ on the regions of the polygon where vertices are closely clustered.
- More vertices in areas of the polygon where vertices are clustered further accentuates the weighting on these regions of the polygon.

To solve the problem of the skewed emphasis of the weighted least-squares error, the weighting $w(k_m)$ for each vertex of the original measured fire perimeter before the interpolation is skew compensated via

$$\tilde{w}_p(k_m) = w(k_m)w_c(k_m)w_u(k_m), \quad w(k_m) = \frac{1}{r(k_m)} \tag{2.3}$$

where $w_c(k_m)$ is a concentration weighting for each vertex used to account for clustering of vertices on the closed polygon and the weighting $w_u(k_m)$ is the user-defined weighting for each vertex, used to actually emphasize certain vertices on the closed polygon. The weighting $w_c(k_m)$ is defined as

$$w_c(k_m) = \frac{1}{m(k_m)} \tag{2.4}$$

where $m(k_m)$ is the number of successive vertices around the $k_{m_{th}}$ vertex with a small adjacent distance $l(k_m)$ that is defined by the relative distance condition

$$\frac{l(k_m)}{l_{mean}} < 0.2, \quad l_{mean} = \frac{1}{N_m}\sum_{k_m=1}^{N_m} l(k_m)$$

where $l(k_m)$ was defined in (2.1). The weighting $w_u(k_m)$ is defined to be 0 for the barrier points, defined as the vertices where the fire perimeter has not changed, and 1 for the other vertices.

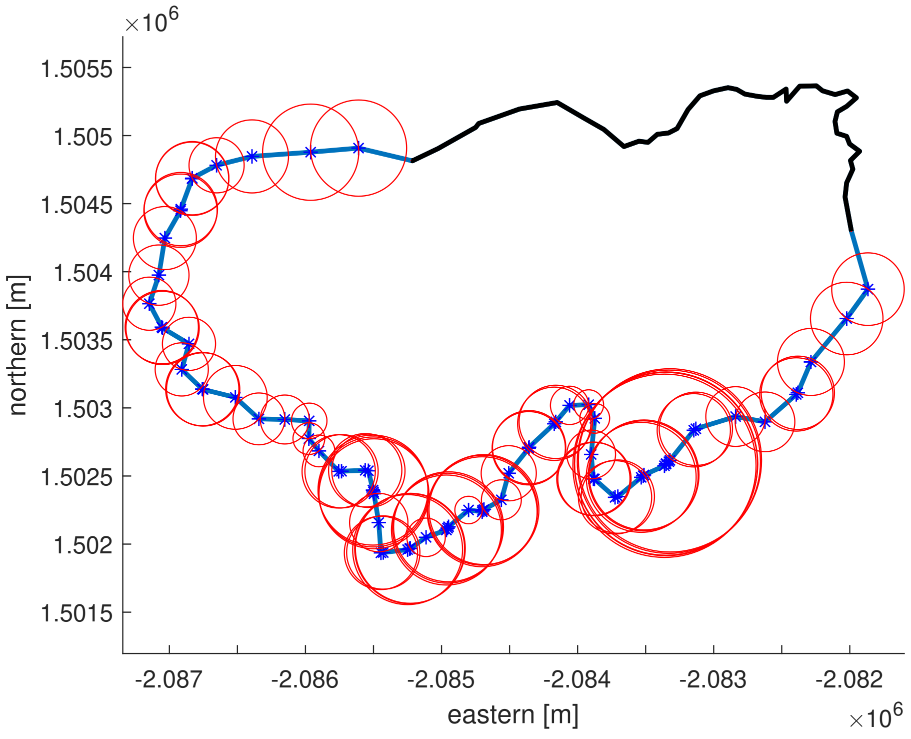


Figure 2.2. Weighting radii $1/\tilde{w}_p(k_m)$ (red circles) for skew compensated least-squares compensation on the vertices (blue stars) and barrier points (black line) of a closed polygon of a measured fire perimeter.

An illustration of the skew compensation is show in Fig. 2.2. On account of the fact that barrier points will not move with the spread of the wildfire, a zero value weighting is assigned to each barrier point. Hence, the weighting radii of barrier points are infinitely large, and not included in Fig. 2.2.

Finally, to also address the re-ordering of the vertices of the closed polygon, consider the short-hand notation based on complex numbers

$$\begin{aligned} x(k) &= e_s(k) + j \cdot n_s(k), \quad k = 1, 2, \ldots, N \\ y(k,q) &= e_m(k) + j \cdot n_m(k), \quad k = q, q+1, \ldots, N, 1, \ldots, q-1 \end{aligned} \tag{2.5}$$

where $x(k) \in \mathbb{C}$ for $k = 1, 2, \ldots, N$ represents the 2D coordinates of vertices of a closed polygon of a simulated fire perimeter starting at index $k = 1$ and $y(k,q) \in \mathbb{C}$ represents the 2D coordinates of vertices of a closed polygon of a measured (and possibly interpolated) fire perimeter, but reordered to start at index q. The ability to adjust the starting point $k = q$ of the closed polygon now allows for the definition of the skew compensated weighted least-squares error

$$s_1 = \min_q \frac{1}{N} \sum_{k=1}^{N} \tilde{w}_p(k)^2 \left| y(k,q) - x(k) \right|^2 \qquad (2.6)$$

where $\tilde{w}_p(k)$ is the interpolated $\tilde{w}_p(k_m)$ defined in (2.3). The starting point $k = q$ is used to remove the dependency of cyclical ordering of complex points describing the closed polygon.

2.3.2 Weighted Surface Area Error

Surface Area of a Closed Polygon

Consider a 2D polygon of a measured fire perimeter given by the N coordinates of the 2D vertices

$$\begin{bmatrix} e(k) \\ n(k) \end{bmatrix}, \quad k = 0, 1, 2, \ldots, N-1$$

ordered by the index k. For the following derivation in this section, k starts from 0 instead of 1, and for an index $k = N$, it is obtained that

$$\begin{bmatrix} e(N) \\ n(N) \end{bmatrix} = \begin{bmatrix} e(0) \\ n(0) \end{bmatrix} \qquad (2.7)$$

formally making the 2D polygon a *closed* polygon. For such a closed polygon, the total surface area S can be computed by taking a signed sum of the surface area of triangular sub-polygons as follows. Consider a triangular 2D sub-polygon that consists of 3 vertices

$$\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} e(k) \\ n(k) \end{bmatrix} \text{ and } \begin{bmatrix} e(k+1) \\ n(k+1) \end{bmatrix}, \quad k = 0, 1, 2, \ldots, N-1 \tag{2.8}$$

which will have a surface area that can be computed by

$$\frac{1}{2} \left| \begin{bmatrix} e(k) \\ n(k) \end{bmatrix} \times \begin{bmatrix} e(k+1) \\ n(k+1) \end{bmatrix} \right|, \quad k = 0, 1, 2, \ldots, N-1 \tag{2.9}$$

where again the property of a closed polygon in (2.7) is used in case $k = N-1$. In (2.9), the symbol $\times$ denotes the cross product and $|\cdot|$ denotes the length of a (cross product) vector. For the 2D vertices, the computation simplifies to

$$\frac{1}{2} \left| e(k)n(k+1) - n(k)e(k+1) \right|, \quad k = 0, 1, 2, \ldots, N-1$$

by writing out the cross product in terms of the $(e(k), n(k))$ and $(e(k+1), n(k+1))$ coordinates. Let $T(k) = e(k)n(k+1) - n(k)e(k+1)$. According to the shoelace formula, or surveyor's area formula [36], if the polygon is counterclockwise oriented, which means the direction from $(e(k), n(k))$ to $(e(k+1), n(k+1))$ is counterclockwise, then $T(k)$ is positive when the origin point $(0,0)$ is on the left side of the edge (facing towards $(e(k+1), n(k+1))$ from $(e(k), n(k))$). Correspondingly, $T(k)$ is negative when the origin point is on the right side of the edge. Therefore, when the 2D closed polygon is oriented counterclockwise, the area of the polygon S can be expressed by the signed sum of the surface area of triangular sub-polygons as follows:

$$S = \sum_{k=0}^{N-1} \frac{1}{2} \left(e(k)n(k+1) - n(k)e(k+1) \right). \tag{2.10}$$

Expectation and Variance

Now let the subsequent vertices at index k and $k+1$ not be given by a single 2D point, but given by a normal probability distribution

$$\begin{bmatrix} \bar{e}(k) \\ \bar{n}(k) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} e(k) \\ n(k) \end{bmatrix}, P(k) \right) \quad \text{and} \quad \begin{bmatrix} \bar{e}(k+1) \\ \bar{n}(k+1) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} e(k+1) \\ n(k+1) \end{bmatrix}, P(k+1) \right) \tag{2.11}$$

where $P(k) > 0$ and $P(k+1) > 0$ denote the covariance matrix of the vertices. The covariance matrix is used to model the (joint) probability between the $e(k)$- and $n(k)$-coordinates of each vertex. Assume that different vertices are independent with each other and $\bar{e}(k)$ and $\bar{n}(k)$ are uncorrelated (the uncertainty on a vertex is expressed by a circle around each vertex), then

$$P(k) = \begin{bmatrix} \sigma_e^2(k) & 0 \\ 0 & \sigma_n^2(k) \end{bmatrix} \quad P(k+1) = \begin{bmatrix} \sigma_e^2(k+1) & 0 \\ 0 & \sigma_n^2(k+1) \end{bmatrix} \tag{2.12}$$

where $\sigma_e(k)$ and $\sigma_n(k)$ are the standard deviations of $\bar{e}(k)$ and $\bar{n}(k)$ respectively. Inspired by [38], the expectation and the variance of $\bar{e}(k)\bar{n}(k+1) - \bar{n}(k)\bar{e}(k+1)$ for each triangular sub-polygon can be calculated based on $T(k)$ as

$$\begin{aligned} \mathbf{E}\left[\bar{e}(k)\bar{n}(k+1) - \bar{n}(k)\bar{e}(k+1)\right] &= \mathbf{E}\left[\bar{e}(k)\right]\mathbf{E}\left[\bar{n}(k+1)\right] - \mathbf{E}\left[\bar{n}(k)\right]\mathbf{E}\left[\bar{e}(k+1)\right] \\ &= e(k)n(k+1) - n(k)e(k+1). \end{aligned} \tag{2.13}$$

The variance of $\bar{e}(k)\bar{n}(k+1)-\bar{n}(k)\bar{e}(k+1)$ is

$$
\begin{aligned}
&\mathbf{Var}\left[\bar{e}(k)\bar{n}(k+1)-\bar{n}(k)\bar{e}(k+1)\right]\\
=&\mathbf{E}\left[\Big(\bar{e}(k)\bar{n}(k+1)-\bar{n}(k)\bar{e}(k+1)-\mathbf{E}\left[\bar{e}(k)\bar{n}(k+1)-\bar{n}(k)\bar{e}(k+1)\right]\Big)^2\right]\\
=&\mathbf{E}\left[\bar{e}(k)^2\bar{n}(k+1)^2\right]+\mathbf{E}\left[\bar{n}(k)^2\bar{e}(k+1)^2\right]\\
+&\mathbf{E}\left[e(k)^2n(k+1)^2\right]+\mathbf{E}\left[n(k)^2e(k+1)^2\right]\\
-&2\mathbf{E}\left[\bar{e}(k)\bar{n}(k+1)\bar{n}(k)\bar{e}(k+1)\right]-2\mathbf{E}\left[\bar{e}(k)\bar{n}(k+1)e(k)n(k+1)\right]\\
+&2\mathbf{E}\left[\bar{e}(k)\bar{n}(k+1)n(k)e(k+1)\right]+2\mathbf{E}\left[\bar{n}(k)\bar{e}(k+1)e(k)n(k+1)\right]\\
-&2\mathbf{E}\left[\bar{n}(k)\bar{e}(k+1)n(k)e(k+1)\right]-2\mathbf{E}\left[e(k)n(k+1)n(k)e(k+1)\right]\\
=&\mathbf{E}\left[\bar{e}(k)^2\bar{n}(k+1)^2\right]+\mathbf{E}\left[\bar{n}(k)^2\bar{e}(k+1)^2\right]-e(k)^2n(k+1)^2-n(k)^2e(k+1)^2\\
=&\left(e(k)^2+\sigma_e^2(k)\right)\left(n(k+1)^2+\sigma_n^2(k+1)\right)\\
+&\left(n(k)^2+\sigma_n^2(k)\right)\left(e(k+1)^2+\sigma_e^2(k+1)\right)\\
-&e(k)^2n(k+1)^2-n(k)^2e(k+1)^2.
\end{aligned}
\tag{2.14}
$$

The expectation and variance of the surface area of the whole closed polygon can then be calculated as

$$\mathbf{E}(S)=\frac{1}{2}\sum_{k=0}^{N-1}\big(e(k)n(k+1)-n(k)e(k+1)\big) \tag{2.15}$$

$$
\begin{aligned}
\mathbf{Var}(S)=\sum_{k=0}^{N-1}\Bigg[&\left(e(k)^2+\sigma_e^2(k)\right)\left(n(k+1)^2+\sigma_n^2(k+1)\right)\\
&+\left(n(k)^2+\sigma_n^2(k)\right)\left(e(k+1)^2+\sigma_e^2(k+1)\right)\\
&-e(k)^2n(k+1)^2-n(k)^2e(k+1)^2\Bigg]
\end{aligned}
\tag{2.16}
$$

For further simplifying the calculation [35], Equation (2.15) can be transformed as follows. By defining $e(N+1)=e(1)$ and $n(N+1)=n(1)$, it can be observed that

$$\begin{aligned}
\mathbf{E}(S) &= \frac{1}{2}\sum_{i=0}^{N-1}\left[e(k)n(k+1)-n(k)e(k+1)\right] \\
&= \frac{1}{2}\sum_{i=0}^{N-1} e(k)n(k+1) - \frac{1}{2}\sum_{i=0}^{N-1} e(k+1)n(k) \\
&= \frac{1}{2}\sum_{i=1}^{N} e(k)n(k+1) - \frac{1}{2}\sum_{i=1}^{N} e(k)n(k-1) \\
&= \frac{1}{2}\sum_{i=1}^{N} e(k)\left(n(k+1)-n(k-1)\right)
\end{aligned} \tag{2.17}$$

For the third equality in (2.17), $e(0)n(1) = e(N)n(N+1)$ is applied in the first sum and index shifting is used in the second sum.

Weighted Surface Area Error

With the simulated fire perimeter $(e_s(k_s), n_s(k_s))$, $k_s = 0, 1, \ldots, N_s - 1$, and the measured fire perimeter $(e_m(k_m), n_m(k_m))$ with an uncertainty mentioned in Section 2.2.1 on each vertex $k_m = 0, 1, \ldots, N_m - 1$. A set of new closed polygons with vertices $(e_d(k_d), n_d(k_d))$, $k_d = 0, 1, \ldots, M_d - 1$ can be obtained by finding the union minus the intersection of the simulated fire polygon and the measured fire polygon. Assume that the number of the newly created polygons is L, and the numbers of vertices included in each polygon are M_d, with $d = 1, 2, \ldots, L$. The weighted surface area S_d^w of the closed polygon with vertices $(e_d(k_d), n_d(k_d))$, $k_d = 0, 1, \ldots, M_d - 1$ can be expressed by

$$S_d^w = \frac{\mathbf{E}(S_d)^\gamma}{\mathbf{Var}(S_d)^{(1-\gamma)}} \tag{2.18}$$

where γ and $1-\gamma$ are the weightings added on the expected value and the variance of the surface area respectively, and $\mathbf{E}(S_d)$ and $\mathbf{Var}(S_d)$ can be calculated by (2.17) and (2.16) respectively. With the assumption that the uncertainty on a vertex is a circle around the vertex and there are only uncertainties on the measured fire perimeter, it can be achieved that

$$\sigma_e(k_d) = \sigma_n(k_d) = \begin{cases} 0, & \text{if } (e_d(k_d), n_d(k_d)) \neq (e_m(k_m), n_m(k_m)), \\ 1/\tilde{w}_s(k_m), & \text{if } (e_d(k_d), n_d(k_d)) = (e_m(k_m), n_m(k_m)), \end{cases} \tag{2.19}$$
$$\tilde{w}_s(k_m) = w(k_m) w_c(k_m)$$

where $\sigma_e(k_d)$, $\sigma_n(k_d)$ are the standard deviations defined in (2.12), and $w(k_m)$, $w_c(k_m)$ are established in (2.3) with no interpolation. Based on (2.18), the weighted error of the whole surface area of the union minus the intersection of the simulated fire perimeter and the measured fire perimeter can be defined as

$$s_2 = \sum_{d=1}^{L} S_d^w = \sum_{d=1}^{L} \frac{\mathbf{E}(S_d)^\gamma}{\mathbf{Var}(S_d)^{(1-\gamma)}}. \tag{2.20}$$

With the definition of $\tilde{w}_s(k_m)$ in (2.19), γ is recommended to be chosen as a value less than or equal to 0.1. Smaller weighting is put on the variance to avoid the erroneous results. For example, if the vertices with extremely large uncertainties are assigned to all the polygons created by the union minus the intersection of the simulated fire polygon and measured fire polygon, then the weighted surface area error is close to zero, and the corresponding simulated fire polygon will be chosen as the optimal simulation that makes no sense. In this chapter, γ is picked as 0.1.

2.4 Wind Condition Estimation with FARSITE

2.4.1 Forward Simulations

In this study, FARSITE is used for the forward simulation of the simulated fire perimeter $x(k)$ as a function of the single wind vector u. FARSITE can be considered as a non-linear mapping $\rho(\cdot)$ for fire progression, simplified to

$$x(k) = \rho(p(k), u, \theta, \Delta_T) \tag{2.21}$$

where the input $p(k) \in \mathbb{C}^{N_p}$ is a closed polygon of N_p vertices representing the initial fire perimeter. The simulated output $x(k) \in \mathbb{C}^{N_x}$, defined earlier in (2.5), is the closed polygon of $k = 1, 2, \ldots, N_x$ vertices representing a simulated fire perimeter obtained after a time step of Δ_T. The additional inputs u represents the single wind vector, and θ denotes a parameter representing fuel content, fuel moisture and topography, all assumed to be constant over the time step of Δ_T.

Unknown wind conditions influence the interpolated and re-ordered vertices of the measured fire perimeter represented by the closed polygon $y(k,q)$ defined in (2.5). The two-dimensional single wind vector u in terms of wind speed and wind direction will also influence the vertices of the simulated fire perimeter represented by the closed polygon $x(k)$ and the weighted surface area S_d^w in (2.18). Along with the definition of the weighting $\tilde{w}_p(k_m)$ in (2.3), $\tilde{w}_s(k_m)$ in (2.19), and γ in (2.18), it is expected that a minimization of s_1 in (2.6), and s_2 in (2.20) as a function of u will provide the best single wind vector to minimize the distance between $x(k)$ and $y(k)$, and surface area of the subtraction between the union and the intersection of the simulated fire polygon and the measured fire polygon, respectively.

2.4.2 Wind Speed and Wind Direction Optimization

The formal problem of finding an estimate of the single wind vector on the basis of a wildfire measurement $y(k)$ using skew compensated weighted least-squares error can be stated as the optimization

$$\begin{gathered}\min_u s_1(u), \quad s_1(u) = \min_q \frac{1}{N} \sum_{k=1}^{N} \tilde{w}_p(k)^2 |y(k,q) - x(k)|^2 \\ x(k) = \rho(p(k), u, \theta, \Delta_T)\end{gathered} \tag{2.22}$$

where $\tilde{w}_p(k)$ is the interpolated $\tilde{w}_p(k_m)$ defined in (2.3) and $y(k,q)$ is defined in (2.5). Similarly, with weighted surface area error, the formal problem can be stated as the optimization

$$\min_u s_2(u), \quad s_2(u) = \sum_{d=1}^{L} \frac{\mathbf{E}(S_d)^\gamma}{\mathbf{Var}(S_d)^{(1-\gamma)}}. \tag{2.23}$$

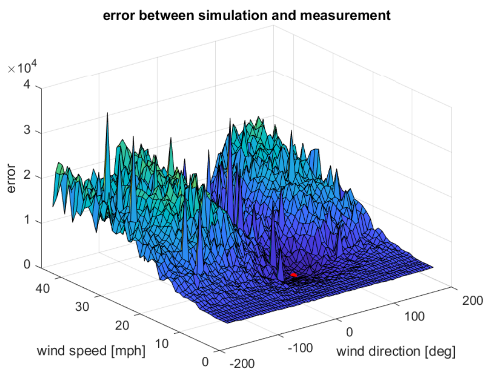
where γ is defined in (2.18), and $\mathbf{E}(S_d)$ and $\mathbf{Var}(S_d)$ are defined in (2.17) and (2.16). Due to the non-linearity and non-convex mapping of $\rho(\cdot)$, a non-linear and iterative optimization is required, typically using the sensitivity or the gradient.

For FARSITE that is responsible for the mapping in (2.21), the sensitivity or gradient $\frac{\partial}{\partial u}\rho(p(k),u,\theta,\Delta_T)$ is unknown. Numerical evaluation of the gradient is computationally expensive and moreover, FARSITE is known to produce occasional erroneous results at some single wind vectors due to numerical problems in interpolation and reconstruction of the main fire perimeter (as will be shown later). These reasons motivate the use of a gradient-free optimization via a grid search and the 2 dimensional size of u motivates a simple 2D gridding procedure over which $s_1(u)$ in (2.22) and $s_2(u)$ in (2.23) are evaluated. The 2D grid of u can be updated and refined iteratively to improve the accuracy of the final optimized solution for u. The pseudo-code for the iterative optimization of $s_1(u)$ and $s_2(u)$ are summarized in Algorithm 2. If the skew compensated weighted least-squares error is chosen, let $s(u_{i,j})$=$s_1(u_{i,j})$; if the weighted surface area error is chosen, let $s(u_{i,j})$=$s_2(u_{i,j})$.

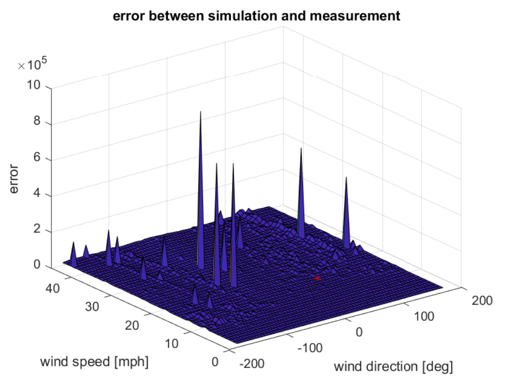
Algorithm 2. Optimizing algorithm

```
Input: $\theta$, $p(k)$, $y(k_m)$, $\Delta_T$, minimum wind condition perturbation $\lambda$ and stopping criterion $\varepsilon$.
Output: Optimized $u \in \mathbb{R}^{2\times 1}$
1: Create $n^2$ points of a symmetric 2D grid $u_{i,j}$ over a desired range $i = 1,2,\dots,n$ and $j = 1,2,\dots,n$ around an initial estimate $u_0$ of the single wind vector.
2: Parallel simulation in FARSITE with $p(k)$, $u_{i,j}$, $\theta$ and $\Delta_T$ to obtain $x_{i,j}(k)$ for each grid point.
3: Compute the $n^2$ weighted error $s(u_{i,j})$ over the grid $i = 1,2,\dots,n$ and $j = 1,2,\dots,n$, or
4: Find the smallest value $\hat{i},\hat{j} = \min_{i,j} s(u_{i,j})$ to select the optimized single wind vector $u_{\hat{i},\hat{j}}$
5: Set $u_0 = u_{\hat{i},\hat{j}}$ and stop when $|s(u_0+\lambda)-s(u_0)| \leq \varepsilon$ or go back to step 1 to refine grid around $u_0$.
```

The skew compensated weighted least-squares error and weighted surface area error are



(a) Results by using skew compensated weighted least-squares error



(b) Results by using weighted surface area error

Figure 2.3. Skew compensated weighted least-squares error $s_1(u_{i,j})$ and weighted surface area error $s_2(u_{i,j})$ at one particular time stamp. The optimal single wind vector of $s_1(u_{i,j})$ and $s_2(u_{i,j})$ are indicated with a red dot.

used to determine the difference between the simulated polygon and the measured polygon of wildfire. Simulations can be performed in parallel to speed up the process of finding the optimal initial single wind vector with the above mentioned algorithm.

2.5 Numerical Results

2.5.1 Maria Fire

The Maria Fire ignited in the evening hours of Thursday, October 31, 2019 and consumed well over 4,000 acres (16 km^2) within its first several hours of burning. The optimization of the single wind vector is performed for this fire at four different time stamps where measurements of the fire perimeter were available. The objective of the optimization is to improve the fire simulations of the fire perimeters with FARSITE in comparison with the observations obtained at four time stamps.

First we illustrate the results of the gradient-free optimization via a grid search summarized in Algorithm 2 in Fig. 2.3. The numerical evaluation of the skew compensated weighted least-squares error $s_1(u_{i,j})$ and weighted surface area error $s_2(u_{i,j})$ over a 2D grid $u_{i,j}$ with wind speed from 0 to 45 mph and wind direction from -180 degrees to 175 degrees in Fig. 2.3 clearly shows the non-differential behavior of $s_1(u)$ and $s_2(u)$, motivating the use of a gradient-free optimization via a grid search. Sporadic large values of $s(u_{i,j})$ for certain single wind vector $u_{i,j}$ are explained by erroneous results due to numerical problems in interpolation and reconstruction of the main fire perimeter by FARSITE, as illustrated in Fig. 2.4. The simulation results show very similar fire perimeters for two single wind vectors that are pretty close to an erroneous result.

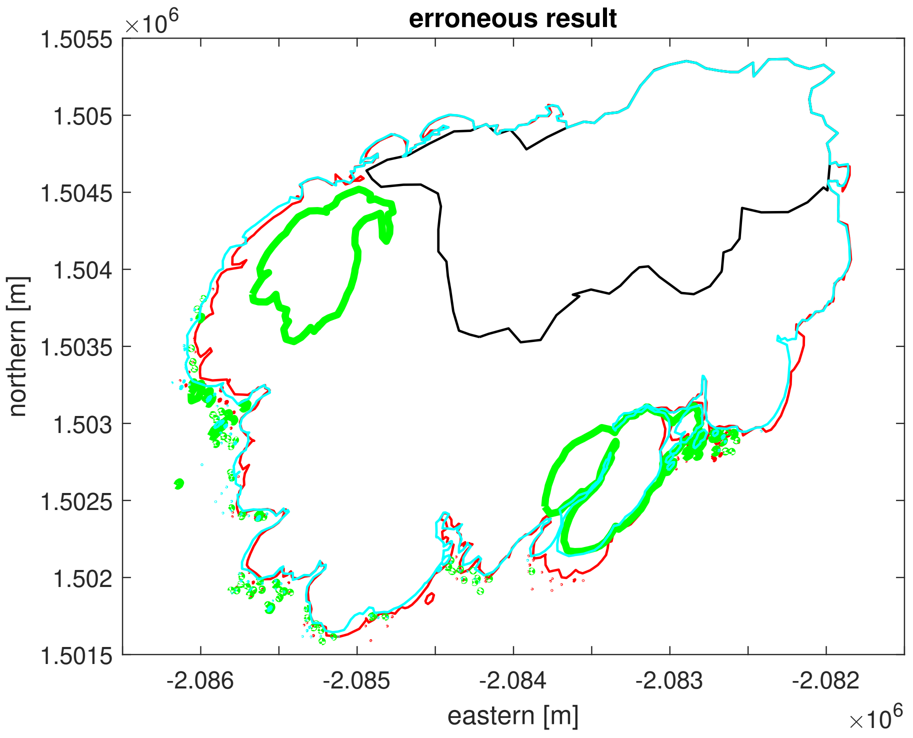


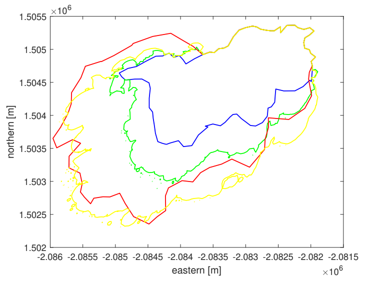
Figure 2.4. Simulations of fire perimeter $x(k)$ with same wind speed of 21 mph, and different wind directions. Wind directions of 34, 35 and 36 deg are represented by red, green, cyan lines. Initial fire perimeter $p(k)$ (black).

Based on gradient-free optimization via a grid search summarized in Algorithm 2, the optimization can correct wildfire simulations when the initial guesses of the single wind vectors are not accurate. Correction of the wildfire simulations with two expressions of error for the four different time stamps where measurements of the Maria Fire perimeter were available are summarized in Fig. 2.5. For each time stamp, the simulated fire perimeter (green lines) based on an initial estimate u_0 of the single wind vector obtained from a weather station can be improved (yellow lines) by the optimization of the single wind vector via Algorithm 2. It can be observed that the optimized single wind vectors provide simulations (yellow lines) that are closer to the measurements (red lines).

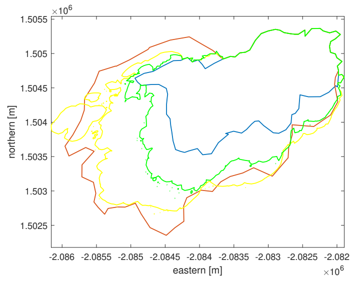
2.5.2 Cave Fire

Although the accuracy of the simulation is improved by using the optimized single wind vector, there are still some parts of the optimized simulation that are somewhat far from the measurement. One reason may be the measurement accuracy, as the combination of aerial surveillance and manually mapped observations is likely to introduce measurement errors. It can also be observed that as the fire perimeter becomes large enough, using only one prevailing wind direction is inadequate for the precise prediction of the wildfire propagation as wind flow is shaped by topography and atmospheric interaction.

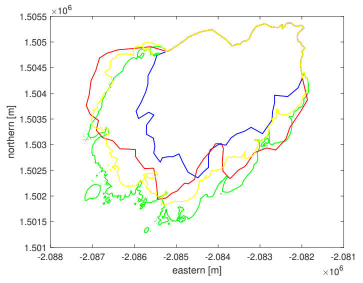
The measurement data available for the Cave Fire included here can better demonstrate the two issues of measurement errors and the assumption of a single wind vector. The 2019 Cave Fire started on November 25 and burned a total of 3,126 acres before being contained on December 19. As shown in Fig. 2.6(a), the top part of the first measurement (after the initial ignition) can be assumed to be wrongly characterized when compared to the second measurement. To be able to account for such errors on the measurement, the weighting $\tilde{w}_p(k_m)$ defined in (2.3) on the vertices in the top part of the first measurement are adjusted to be zero for the skew compensated weighted least-squares error. The effect of the weighting radii is illustrated in the image on the left in Fig. 2.6(b)). Due to the fact that the measurements in the top part of the first measurement are weighted with 0 for the skew compensated weighted least-squares error, the corresponding weighting radii are approaching infinity that is not included in the figure. However, the remaining points of the measurement are still allowed to be used for optimization of the single wind vector at this time stamp. For the weighted surface area, on account of the limitation that multiple uncertainties of the vertices will finally act only on one weighted surface



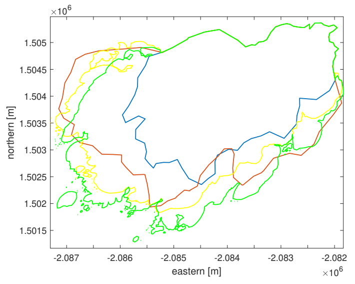
(a) 7:37 p.m. (using $s_1(u)$)



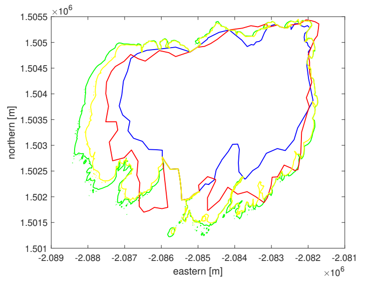
(b) 7:37 p.m. (using $s_2(u)$)



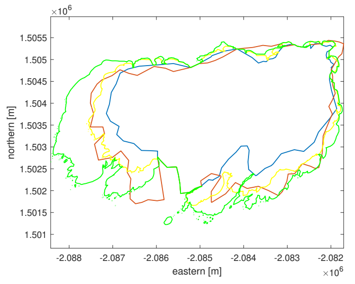
(c) 7:58 p.m. (using $s_1(u)$)



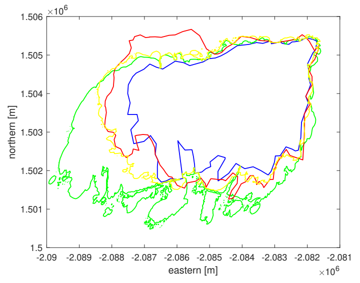
(d) 7:58 p.m. (using $s_2(u)$)



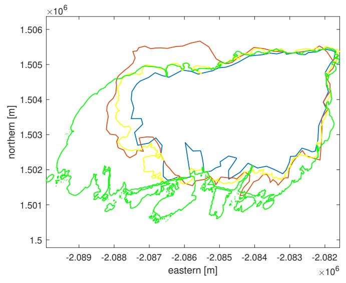
(e) 8:31 p.m. (using $s_1(u)$)



(f) 8:31 p.m. (using $s_2(u)$)

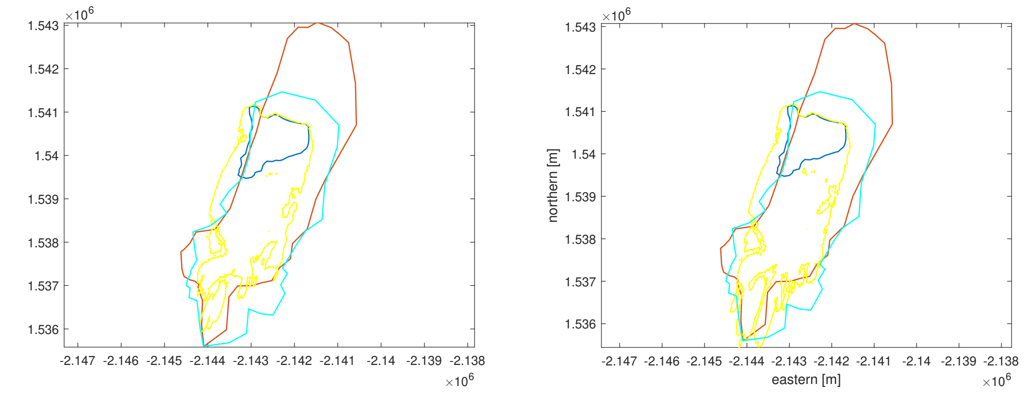


(g) 8:56 p.m. (using $s_1(u)$)

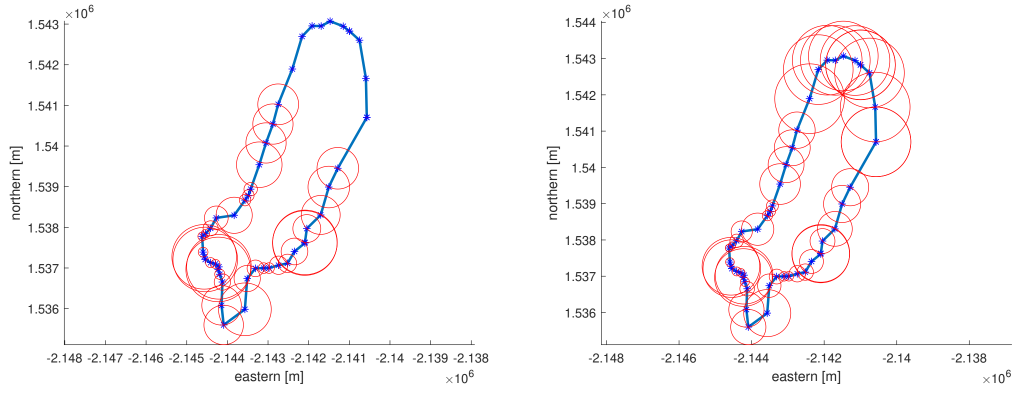


(h) 8:56 p.m. (using $s_2(u)$)

Figure 2.5. Comparison of measured and simulated fire perimeters for u_0 and optimized u by $s_1(u)$ and $s_2(u)$. Initial ignition (blue); Simulation with u_0 (green); Simulation with u (yellow); Measurement at next time step (red).



(a) Initial fire perimeter (blue); Measurements at the first time stamp (red) and the second time stamp (cyan) after the initial ignition. Simulations with optimized single wind vector (yellow).

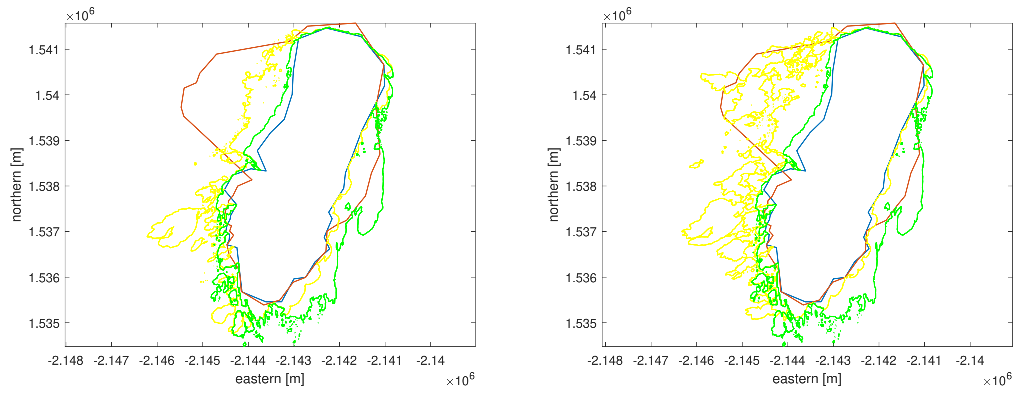


(b) Weighting radii $1/\tilde{w}_p(k)$ (red circles, left) and $1/\tilde{w}_s(k)$ (red circles, right) on the vertices of the measured Cave Fire at 03:48 a.m. (blue stars).

Figure 2.6. Simulation and measurements of the Cave Fire with measurement errors. Skew compensated weighted least-squares error (left). Weighted surface area error (right).

area, the infinitely large weighting radii should not be used in case they makes weighted surface area errors of all polygons in the union minus intersection of the simulated fire polygon and predicted fire polygon zero. Therefore, the weighting radii of the top part of the measurement are adjusted to be the same as the largest weighting radius in the other parts of the measurement to reflect the large uncertainty in the top part.

When the Cave Fire grows to a large dimension, as illustrated in Fig. 2.7, it becomes difficult to match the measured fire perimeter with a simulated fire perimeter via single prevailing wind direction. The gradient-free optimization of Algorithm 2 does a better job covering the east side of the fire, but the west side of the fire cannot be accurately covered with a single wind vector due to the topography and atmospheric wind shear effects acting on the fire. This illustrated the limitations of optimizing only a single wind vector.



(a) Results by using skew compensated weighted least-squares error (b) Results by using weighted surface area error

Figure 2.7. Comparison of measurement and simulation of the Cave Fire with initial ignition at 05:15 a.m. (blue). Simulation with initial (green), and optimized (yellow) single wind vector. Measurement (red).

2.6 Summary

This chapter shows how fire perimeter measurements can be used to improve the accuracy of a wildfire perimeter simulation, by using the measurement to estimate and correct the prevailing wind speed and wind direction for the simulation. The estimation is based on two carefully defined uncertainty weighted errors. The first error characterization is a skew compensated uncertainty weighted least-squares error that provides a direct comparison of the vertices of a simulated and a measured (noisy) wildfire perimeter. The second error characterization is formulated as an uncertainty weighted surface area based on difference between the union and the intersection of the surface of a simulated and a measured (noisy) wildfire perimeter. The uncertainty based weighting in the least-squares error can account for vertex accuracy and be adjusted for a skewed weighting caused by unequally distributed vertices on the closed polygon of the fire perimeter. In both cases, the (skew compensated) uncertainty radii are used to compute a uncertainty weighted error. A gradient-free optimization via a grid search that uses (refined) grid of the two-dimensional single wind vector and exploits parallel computations with FARSITE fire modeling has been done to compute the optimal single wind vector. Numerical results on actual wildfire perimeter data obtained from two recent destructive fires in California confirm the improvement of the accuracy of the wildfire perimeter simulations. The skew compensated weighted least-squares error is adept at flexibility of applying complicated uncertainties on each vertex, and weighted surface area error has an advantage in the simplification of the computational complexity and reduction of computational time. Limitations of the proposed methods are due to the optimization of a single wind vector – an assumption that may not hold when a wild fire covers a large area with varying topographical features.

2.7 Acknowledgement

Chapter 2, in full, is a reprint of the material as it appears in "Estimation of wildfire wind conditions via perimeter and surface area optimization", Tan, L., de Callafon, R. A., Block, J., Crawl, D., Çağlar, T., & Altıntaş, I. (2022). Journal of Computational Science, 61, 101633. The dissertation author was the primary investigator and author of this paper.

Chapter 3

Characterizing Wildfire Perimeter Polygons from QUIC-Fire

QUIC-Fire is a modern fire simulation tool that can simulate the progression of three-dimensional fuel consumption over a landscape, modeling the interaction of a wildfire with weather such as wind conditions around the wildfire. The resulting simulation gives a detailed progression of the consumed three-dimensional fuel that can be eloquently mapped to an image of a burn area in the landscape as the wildfire progresses over time. Although an image of burned vegetation over a landscape gives detailed information of the activity and coverage area of a wildfire, a numerical characterization of the boundary of the burn area can be used for a variety of computations. The boundary of the burn area, also labeled as the wildfire perimeter, can be parametrized with a closed polygon. The set of ordered vertices of the closed polygon provide a compact numerical representation of the location of the wildfire and can be used for computations related to fire coverage area and modern wildfire assimilation techniques to improve the prediction of wildfire progression. Designing a robust algorithm to create a wildfire perimeter in the form of a set of ordered vertices of a closed polygon around the image of consumed vegetation in a landscape is not a trivial task. This chapter discusses the properties of two such algorithms: the iterative minimum distance algorithm (IMDA) and quadriculation algorithm (QA) to obtain a closed polygon for a wildfire perimeter. To illustrate the effectiveness, these two algorithms are applied to multiple image (raster) data of a burn area in the landscape of a wildfire created by QUIC-Fire simulations. It is shown that both algorithms are robust in computing wildfire perimeters, and computational time are less than one second for each image created by QUIC-Fire. As such, this work contributes to the development of computational methods to automate the process of characterizing the closed polygon of a wildfire perimeter based on burn area images.

3.1 Introduction

Vegetation dispersed over a landscape is the main fuel component that drives many wildfires. As a wildfire consumes this fuel under the influence of external wind and other weather conditions, it creates a 'burn area' or 'burn scar' of consumed fuel in the landscape that can cause significant damage, economic loss and environmental impacts. Clearly, understanding the wildland fire behavior and reducing the effects of wildfires by either controlling vegetation via prescribed burns or improving predicting the progression of a wildland fire are desirable.

Improving the prediction of wildfire progression has been an active area of research [4, 5, 39]. Data assimilation by combining wildfire modeling and ensemble Kalman filter is applied in [7, 28, 29], while several studies on the influence of wind condition and fuel have been conducted [22,26,40]. Many fire behavior models have been developed to improve the prediction of the wildfire progression [3, 8, 9, 13, 41], and the focus on controlling wildfires by prescribed burns is driven by QUIC-Fire [13]. QUIC-Fire can serve as a modern fire simulation tool to simulate the progression of three-dimensional (3D) fuel consumption over a landscape, while also approximating the dynamic interaction of a fire with weather including wind conditions in the atmosphere around the fire. QUIC-Fire can also takes into account the interactions between multiple fires and can compute fire progression at the resolution of one meter.

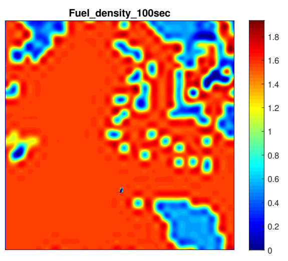
A wildfire perimeter, defined as a closed polygon around the burn area of a wildfire or prescribed burn, is an important numerical characterization of the impact of the fire and can be used for a variety of computations. Most wildfire perimeters are obtained from 2D images [42–44], while the consumed 3D fuel created by QUIC-Fire simulation is mapped to a 2D image of a burn area in the landscape as the wildfire progresses over time. So even for the output of QUIC-Fire, it is desirable to create an algorithm to compute the closed polygon of the wildfire perimeter.

Designing a robust algorithm to create a wildfire perimeter in the form of a set of ordered vertices of a closed polygon around the image of consumed vegetation in a landscape is not a trivial task. Edge detection methods have been applied to wildfire images [45, 46], but only find a set of unordered boundary points that is not suitable to produce a closed polygon. In addition, a wildfire perimeter may include one main closed polygon and multiple additional closed polygons due to sporadic fire spread caused by embers and no assumption can be made on the shapes of the polygons. Due to this complexity of multiple wildfire perimeters, traditional pattern recognition algorithms [47, 48] are not directly applicable.

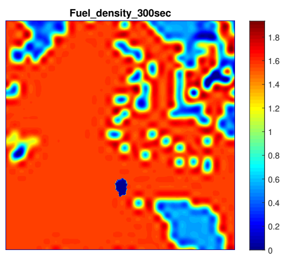
This chapter discusses the properties of two algorithms: the iterative minimum distance algorithm (IMDA) and quadriculation algorithm (QA) to create a closed polygon of a wildfire perimeter. The IMDA is based on continually connecting two closest points in the set of unordered boundary points determined by conventional image edge detection. A threshold value is set up to assist in determining whether two points in a cluster are closely located. If one cluster is far away from other clusters, then it is regarded as a new isolated polygon representing a separate fire perimeter due to embers. From a completely different point of view, the QA creates a polygon by recursively dividing the raster image into indivisible rectangles, where all the internal pixels of the rectangles have the same color, and then merging adjacent rectangles that have the same color. In general, QA avoids the process of ordering the unordered boundary points, but takes a longer time when merging the different polygons.

3.2 QUIC-Fire Output Data

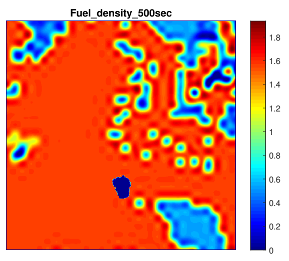
As mentioned in the introduction, the focus of this chapter is to discuss the properties of the iterative minimum distance algorithm (IMDA) and quadriculation algorithm (QA) to obtain closed polygons of wildfire perimeters based on images of consumed vegetation in a landscape. This section summarizes the QUIC-Fire output data used for the evaluation of the IMDA and QA. The QUIC-Fire output data consist of images of the fuel densities over a landscape at ground level (below 10m) at different time stamps (100s, 300s, 500s, 700s, 900s, 1100s) as a prescribed burn or wildfire progresses. The images of the QUIC-Fire output data are given in Figure 3.1.



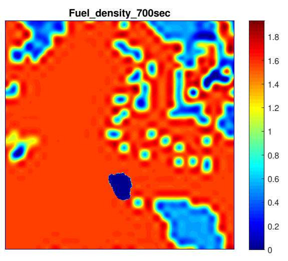
(a) 100s



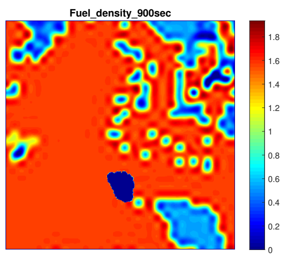
(b) 300s



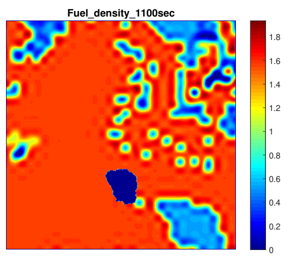
(c) 500s



(d) 700s



(e) 900s



(f) 1100s

Figure 3.1. Fuel densities at different time stamps after the wildfire begins.

From Figure 3.1, it can be observed that as time increases, the dark blue area with near zero fuel density becomes larger, which means the fuels are consumed and the wildfire is spreading. The burn area of the wildfire at different time stamps can then be detected by comparing the difference of the corresponding fuel densities and the fuel density before the wildfire starts, leading to the black/white images given in Figure 3.2.

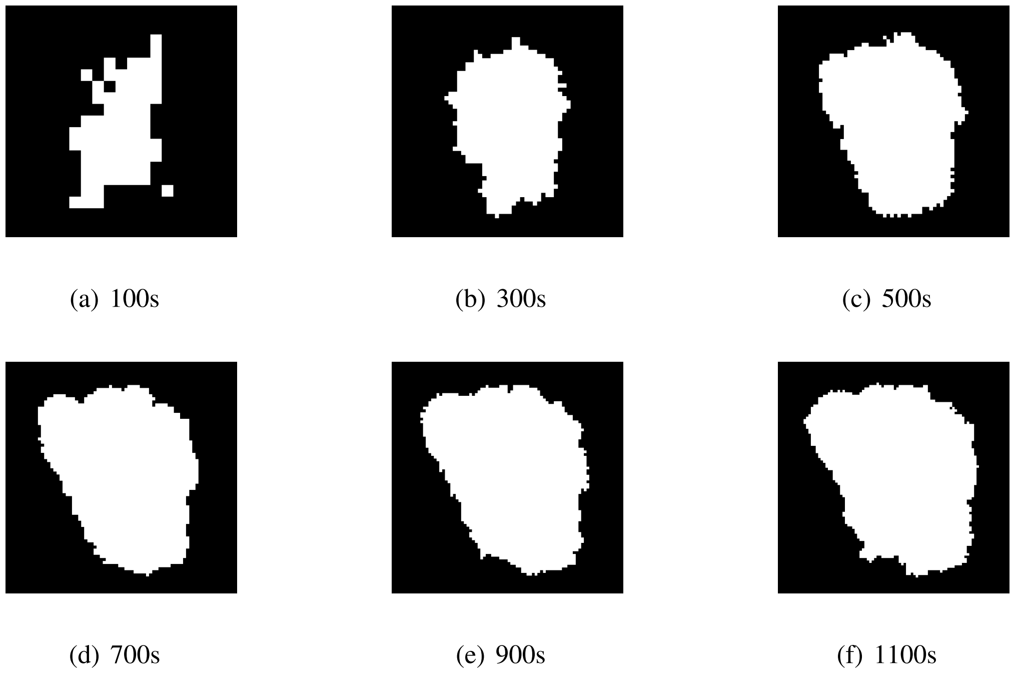


Figure 3.2. Burn area data at different time stamps after the wildfire begins. The white area is the burn area, and the black area is the unburned area. The scales of the six plots are selected differently for a better view.

3.3 Polygon algorithms for wildfire perimeters

3.3.1 Image data

The burn area at six different time stamps are illustrated in Figure 3.2. The white area represents the burn area with $y = 1$, and the black area represents the unburned area with $y = 0$, where

$$y_{i,j} = f([i,j],b) = \begin{cases} 0, & \text{if } b = 0 \\ 1, & \text{if } b > 0 \end{cases} \tag{3.1}$$

In (3.1), $[i, j]$ is a vector providing the position information of the target pixel in the image, and b is the absolute difference value between the fuel densities at the current time stamp and before the wildfire starts. The variable y is used to describe the area (burn area or unburned area) the pixel (at $[i, j]$) belongs to.

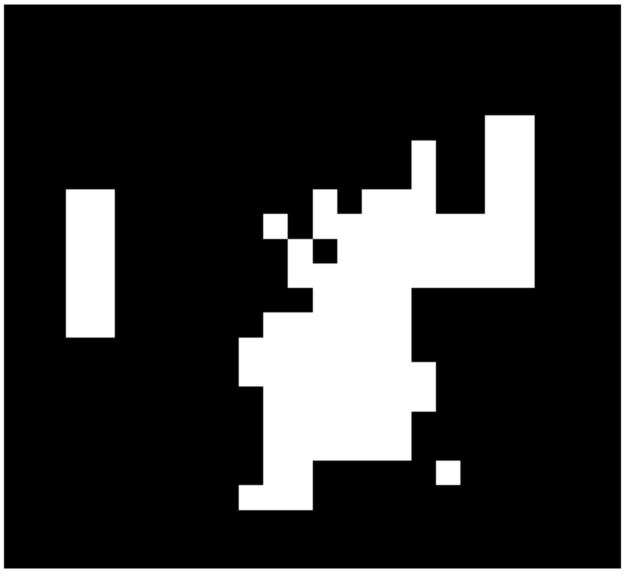


Figure 3.3. Modified output of the QUIC-Fire with extra rectangular burn area. The white area represents the burn area, and the black area represents the unburned area.

To better cover all possible situations of wildfire and illustrate the performances of IMDA and QA, one of the burn area outputs of Figure 3.2 has been increased in complexity by adding a separate (rectangular) burn area, and removing part of the original burn area, as indicated in Figure 3.3. The additional burn area is added to verify if both the IMDA and QA can recognize multiple wildfire perimeters within the data of Figure 3.3.

3.3.2 Quadriculation algorithm

The first method of finding an ordered set of vertices of a closed polygon around a burn area is the quadriculation algorithm (QA). Inspired by fire simulation tool FARSITE [8], the QA solves the problem in two main steps of division and union. Due to the fact that the minimum unit of a rasterized burn area image is a pixel, QA quadriculates the target image into four squares or rectangles recursively until all pixels in one square or rectangle have same value y. The process is illustrated on a simple example in Figure 3.4(a). It can be observed that after the first division,

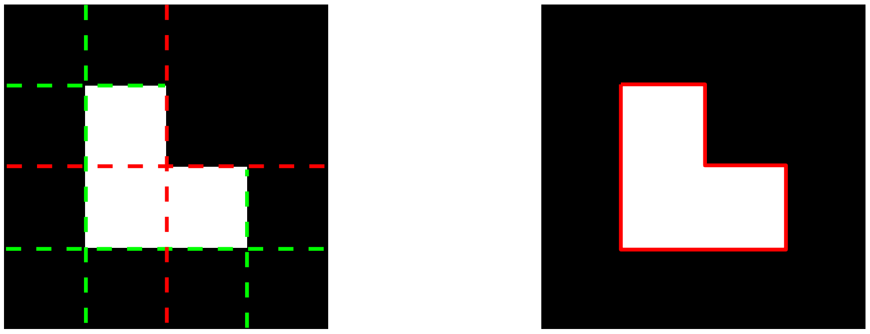

(a) Recursive division. (b) Union.

Figure 3.4. Division and union in QA. The dashed red and green lines represents the first and second division respectively. The red solid line represents the polygon. The white and black area are with $y = 1$ and $y = 0$ respectively.

only the pixels in the right-top square of the image have the same value ($y = 0$). Therefore, another quadriculation is needed. The second division should be applied on left-top, left-bottom, right-bottom squares because pixels have different values $y = 1$ and $y = 0$ in these three squares, and no division should be applied on the right-top square.

After the recursive division, the adjacent squares or rectangles with same value y should be joined, and the perimeter of the polygon in Figure 3.4(a) can be obtained in Figure 3.4(b). With the precision of one meter for QUIC-Fire, the size of each cell is one meter times one meter. Therefore, the polygon obtained by QA can be accurate enough to describe the wildfire perimeter. The process of QA is summarized in Algorithm 3.

Algorithm 3. QA

```
Input: Fire image
Output: Polygons representing wildfire perimeters
1: Recursively quadriculate the image into four squares or rectangles until all pixels in one
   square or rectangle have same value $y$.
2: Join the adjacent squares or rectangles until the pixels in adjacent squares or rectangles have
   different value $y$.
```

The QA is known to take quite some computation time due to two main steps of division and union that scales up as the image size increases. It would be beneficial to have an algorithm that can also handle large images with multiple burn areas. The proposed algorithm is presented in the next section.

3.3.3 Iterative minimum distance algorithm (IMDA)

Preparatory work

The IMDA solves the problem of finding an ordered set of vertices of a closed polygon around a burn area by selecting and ordering the set of unordered boundary points. First, a standard image edge detection algorithm is applied to Figure 3.3 to acquire the boundary points. The boundary points are detected by comparing the value $y_{i,j}$ of the target pixel with its surroundings. An abrupt change in the y value of the pixel expressed by

$$\begin{aligned} &|y_{i-1,j} - y_{i,j}| \neq 0 \text{ and } y_{i-1,j} = 1, \\ \text{or } &|y_{i,j} - y_{i,j-1}| \neq 0 \text{ and } y_{i,j-1} = 1, \\ \text{or } &|y_{i+1,j} - y_{i,j}| \neq 0 \text{ and } y_{i+1,j} = 1, \\ \text{or } &|y_{i,j} - y_{i,j+1}| \neq 0 \text{ and } y_{i,j+1} = 1, \end{aligned}$$

and the pixel at i, j can be regarded as a boundary point. Due to the fact that the precision of the QUIC-Fire data can be as small as one meter, the edge detection achieves a resolution of one meter.

Naive minimum distance

To motivate the IMDA, first consider the simplest method for the rearrangement of the unordered vertices or boundary points: choosing an arbitrary starting point and find the closest point to the previous selected point. In this native minimum distance (NMD) check, an important requirement is to avoid a self-intersection of the polygon.

With the set of the unordered boundary points B, the starting point b_1 is first selected arbitrarily. Then, remove b_1 from the set B, and find a new point b_v ($v > 1$) with the minimum distance to b_1 in B. If the distance between the last two selected points b_{v-1} and b_v, where $v \geq 3$, is larger than the distance from b_{v-1} to b_1, b_{v-1} is connected to b_1 directly to produce a closed polygon. To ensure there is no problem of self-intersection, the NMD checks whether the line segment $b_{v-1}b_v$ intersects with any previous created line segments. If there exists an intersection, the point b_{v-1} is deleted and connect $b_{v-2}b_v$. This process iterates until no intersection exists.

During the process of finding b_v, two or more points can be found with same distance to the previous selected point (multi-choice situation). To solve this problem, each choice will be stored and the corresponding closed polygon is recorded. The polygon with the largest number of vertices is picked as an optimal choice because more vertices means more detailed information. If there are multiple polygons with same number of vertices, more constraints such as the area of the polygon, can be added to select the optimal polygon.

The main problem for the NMD check is that for each multi-choice situation, two or more complete polygons that are generated also need to be stored for comparison purpose. Storing and comparing polygons may be an computationally expensive process, especially when the numbers of boundary points and multi-choice situations increase. This problem is illustrated in a simple case of Figure 3.5. It can be observed that the red polygon better describes the burn area than the cyan dashed polygon, and the only difference between these two polygons is located inside the green dashed rectangle in the figure.

Next to storing and comparing multiple polygons, the NMD check cannot deal with the case when a wildfire has multiple disjoint burn areas to create multiple wildfire perimeters. These problems lead to a modification of the NMD check and result in the actual IMDA.

Computation of ordered vertices of the closed polygon

In the computation of ordered vertices of the closed polygon in IMDA, one initial main polygon is first obtained by arbitrarily choosing a point in the multi-choice situation. All left

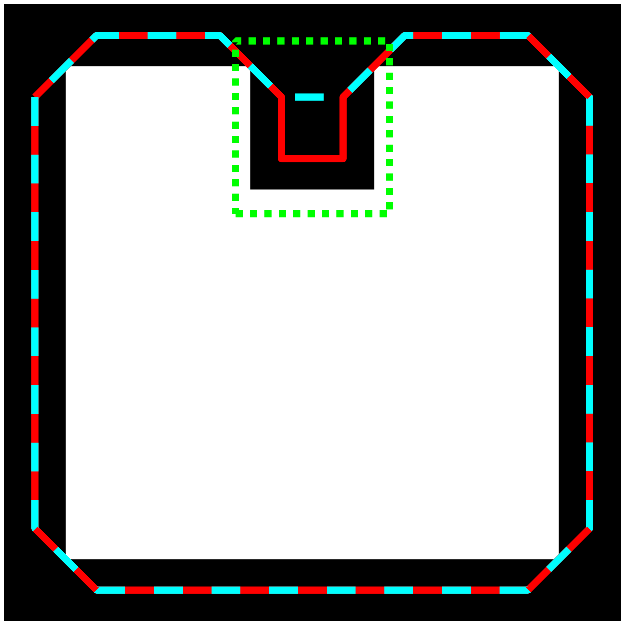


Figure 3.5. Two possible polygons after removing self-intersections (red line and cyan dashed line). The green dashed rectangle shows the two-choice difference.

points are used to modify the initial main polygon or create a new isolated polygon. It is still assumed that all the unordered boundary points can be used only once, but with one more constraint: the largest distance between two adjacent boundary points should be smaller than $d = \sqrt{2}$ due to point-to-point pixel distances. Following this distance observation, there are two main steps in IMDA: the first step is to obtain an initial main polygon, and the second step is to modify the obtained polygon and decide whether there is an extra isolated polygon. The logic of each step is described as follows.

For the first step, an arbitrary starting point b_1 is selected from the set B of the unordered boundary points to be the first point of the set P that is used to restore the ordered vertices of the polygon of a wildfire perimeter. The point with the minimum distance to the previously selected point in P is chosen from B and added to P one by one. If there are multiple points with the same minimum distance to the previously selected point, the first point in order is selected. During the selection, if no other points in set B have the distance smaller than d with respect to the last point in P, the distance from the last point in P to the starting point b_1 is checked. If the distance is smaller than d, a closed polygon is created. On the contrary, if the distance is larger than d, it means the current trajectory is not correct. Therefore, the last point in P needs to be deleted and moved to a different set B_c so that this point can be reused again and ordered correctly. Repeat deleting the last point in P and move it to set B_c until a point with a distance smaller than d to the updated last point of P can be found in B, or the updated last point of P has the distance smaller than d to the starting point b_1. The first step is finished by creating an initial closed polygon.

With all the points moved from B_c to B, and clearing the set B_c, the second step is initiated by finding the nearest point in B to any vertex in P, if the distance is larger than d, it means no improvement can be achieved by the initial main polygon, and an isolated polygon exists. Hence, the first step should be repeated for the updated B to create a new initial polygon. If there exists a point in B with a distance to the nearest vertex in P smaller than d, it means the initial closed polygon can be updated. Based on closest vertex in P as the first point of the trajectory P_c, the nearest point from set B to the last point in P_c is found. If the distance from the newly detected point in B to the last point in P_c is smaller than d, then add the newly detected point to the set P_c. If no more points in B has the distance smaller than d to the last point of P_c, find the closest point in P to the last point in P_c. If the distance from the closest point in P to the last point in P_c is smaller than d, add the detected closest point in P to the set P_c. If the distance is larger than d, delete the last point in P_c, and add it to B_c until the distance from the closest point in P to the last point in P_c is smaller than d. Then add the detected closest point in P to the set P_c.

One important thing to note here is that the first point and the last point in P_c should be different from each other. Based on the first point and the last point of P_c, add P_c to the initial created polygon. If the previously created polygon has other vertex between the first point and the last point of P_c, which means connecting P_c to P will lead to the deletion of previously selected vertices. Then, whether connecting P_c to P depends on whether connecting P_c will increase the area of the polygon. If connecting P_c to P can increase the area of the polygon, P_c is connected to P and replace the corresponding part selected in the first step. Otherwise, keep P as it is. Iterate this process until there is no point left in B and B_c. The logic process of the IMDA is summarized in Algorithm 4.

Algorithm 4. IMDA

Input: Unordered boundary points B and threshold value d.

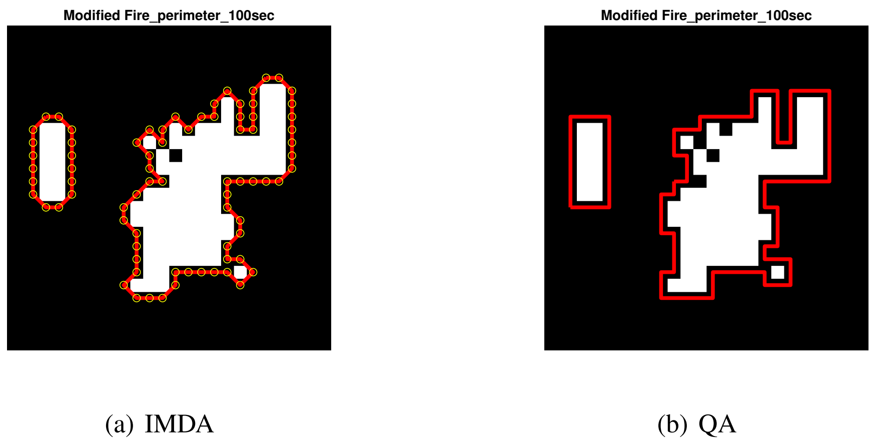
Output: Polygons representing wildfire perimeters

```
1: Pick the arbitrary starting point $b_1$ in $B$, and delete $b_1$ in $B$.
2: Find a closed polygon $P$ based on finding the point with the minimum distance that is smaller
   than $d$ to the previously selected point.
3: Find a trajectory $P_c$ when the distance from any point in $B$ to $P$ is smaller than $d$.
4: If adding $P_c$ to $P$ will not result in the deletion of the previously selected point in $P$, add $P_c$
   to $P$.
5: If adding $P_c$ to $P$ will result in the deletion of the previously selected point in $P$, $P_c$ is added
   to $P$ when it increases the area of the polygon.
6: Iterate steps 3-5 until no points in $B$ have distance smaller than $d$ to $P$.
7: Repeat the above steps if there are multiple polygons.
```

3.4 Numerical Results

IMDA and QA are applied to the modified burn area data of Figure 3.3 to verify the detection of multiple fire perimeters. The resulting closed polygons created by IMDA and QA are shown in Figure 3.6. It is clear that both IMDA and QA produce the two distinct fire perimeters, but it can also be observed that IMDA provides slightly tighter polygons around the burn area as the polygons are not restricted to horizontal and vertical lines as in QA.

To further compare the performance of IMDA and QA, the algorithms are applied to the burn area data of at the six different time stamps of Figure 3.2. The visual results are summarized in Figure 3.7 with the same conclusion: both IMDA and QA produce correct results, but IMDA provides slightly tighter polygons. The more telling observations come from Table 3.1, where it can be seen that the computation time of IMDA scaled favorably compared to QA as the image size and the burn area of the wildfire perimeter increases. As reference for the computation time, all calculations were performed on an Intel Core i7-7500U CPU with 16 GB RAM.

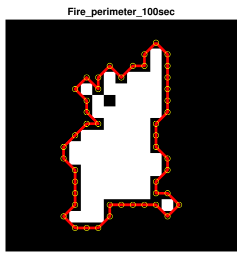

Modified Fire_perimeter_100sec
Modified Fire_perimeter_100sec

(a) IMDA (b) QA

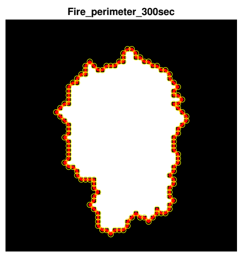
Figure 3.6. polygons of the wildfire perimeter (red lines). Burn area (white), unburned area (black), detected boundary points (yellow circles).

Table 3.1. Computation time of IMDA and QA

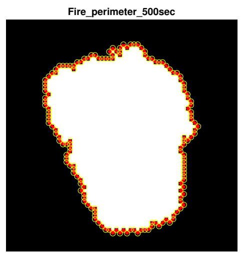
	100s	300s	500s	700s	900s	1100s
IMDA	0.019s	0.019s	0.021s	0.020s	0.023s	0.024s
QA	0.041s	0.132s	0.185s	0.205s	0.362s	0.352s



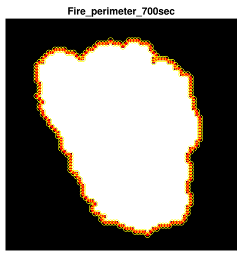
(a) 100s



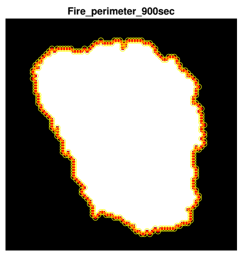
(b) 300s



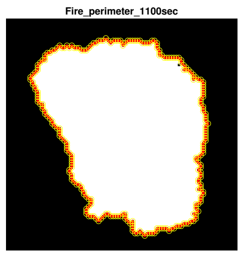
(c) 500s



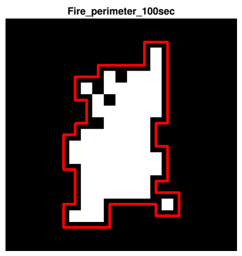
(d) 700s



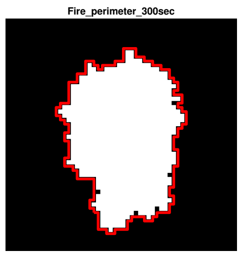
(e) 900s



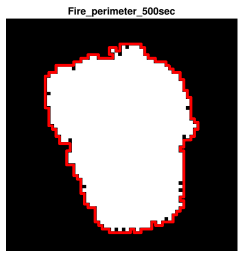
(f) 1100s



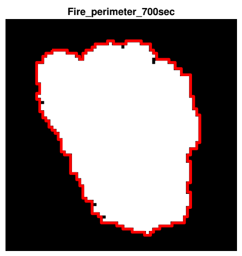
(g) 100s



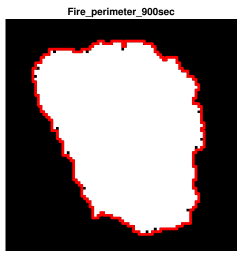
(h) 300s



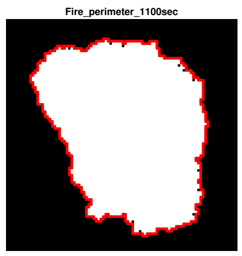
(i) 500s



(j) 700s



(k) 900s



(l) 1100s

Figure 3.7. Polygons of the wildfire perimeter (red lines). Burn area (white), unburned area (black) and detected boundary points (yellow circles).

3.5 Summary

This chapter compares two algorithms, iterative minimum distance algorithm (IMDA) and quadriculation algorithm (QA), to obtain the closed polygons that parametrize wildfire perimeters. The IMDA is based on continually connecting two closest points in the set of unordered boundary points determined by conventional image edge detection. A threshold value is set up to assist in determining whether two points are closely located. From a completely different point of view, the QA creates a polygon by recursively dividing the raster image into indivisible rectangles, where all the internal pixels of the rectangles have the same color. Then, the QA merges adjacent rectangles that have the same color. Using simulation data produced by QUIC-Fire that consist of raster images of consumed vegetation in a landscape, the performance of IMDA and QA is compared. Although the logic of IMDA is more complicated, IMDA produces slightly tighter polygons around the burn area compared to QA, as the polygons are not restricted to horizontal and vertical lines in the image resolution. Moreover, the computation time of IMDA scaled favorably compared to QA as the image size and the burn area of the wildfire perimeter increase.

3.6 Acknowledgement

Chapter 3, in full, is a reprint of the material as it appears in "Characterizing Wildfire Perimeter Polygons from QUIC-Fire", Tan, L., de Callafon, R. A., & Altıntaş, I. (2022, June). In Computational Science–ICCS 2022: 22nd International Conference, London, UK, June 21–23, 2022, Proceedings, Part I (pp. 611-622). Cham: Springer International Publishing. The dissertation author was the primary investigator and author of this paper.

Chapter 4

Wildfire Perimeter Detection via Iterative Trimming Method

The perimeter of a wildfire is essential for prediction of the spread of a wildfire. Real-time information on an active wildfire can be obtained with Thermal InfraRed (TIR) data collected via aerial surveys or satellite imaging, but often lack the actual numerical parametrization of the wildfire perimeter. As such, additional image processing is needed to formulate closed polygons that provide the numerical parametrization of wildfire perimeters. Although a traditional image segmentation method (ISM) that relies on image gradient or image continuity can be used to process a TIR image, these methods may fail to accurately represent a perimeter or boundary of an object when pixels representing high infrared values are sparse and not connected. An ISM processed TIR image with sparse high infrared pixels often results in multiple disconnected sub-objects rather than a complete object. This chapter solves the problem of detecting wildfire perimeters from TIR images in three distinct image processing steps. First, Delaunay triangulation is used to connect the sparse and disconnected high-value infrared pixels. Subsequently, a closed (convex) polygon is created by joining adjacent triangles. The final step consists of an iterative trimming method that removes redundant triangles to find the closed (non-convex) polygon that parametrizes the wildfire perimeter. The method is illustrated on a typical satellite TIR image of a wildfire, and the result is compared to those obtained by traditional ISMs. The illustration shows that the three image processing steps summarized in this chapter yield an accurate result for representation of the wildfire perimeter.

4.1 Introduction

As unprecedented wildfire activity has occurred in recent years, the reliable prediction of the spread of an active wildfire has proven to be challenging task. Ensemble Kalman filtering [49] is often used as the data assimilation technique for wildfire spread prediction [5]. Many applications have combined a data-driven fire model such as FARSITE [8] with ensemble Kalman filtering to improve the prediction accuracy [7, 28, 29]. Such data assimilation techniques do rely on the availability of (past) fire perimeter measurements to predict (future) wildfire perimeters for characterizing wildfire progression.

For the moment, manual delineation is still frequently used to obtain the wildfire perimeter. Due to the significant advances in sensor technology for wildfire monitoring, new real-time data sources such as Thermal InfraRed (TIR) imaging can be used for a data-driven approach to predict wildfire progression. For example, wildfires can be monitored via MODIS data [23], satellite heat images [24] or thermal infrared imaging (TIR) on aerial flight systems [25] that characterize ground temperature. In case of TIR or heat images, image pixels represent temperature information or strength of the infrared band. In particular for an RGB image, the R value is chosen as the data for the image pixels.

Many contributions can be found that provided automated extraction of wildfire perimeters by applying classic edge detection algorithm, such as global intensity thresholding algorithm, Sobel gradient operator, and Canny edge detector [45, 50]. Among them, Canny method has the ability to outperform the others [45]. Such edge detection methods might be good tools to automatically delineate the wildfire perimeter, and the application of other edge detection methods, such as graph-cut method and level set method, are beneficial for the detection of wildfire perimeter. However, due to limited resolution of the image, discontinuity of the two-dimensional image data, and possibly partial activity of a wildfire along its boundary, identification of the most recent fire perimeter remains a challenge. Furthermore, different image pixels may be independent and subjected to noise or temporary fire inactivity. As a result, the burn area in the wildfire image can be disconnected or even sparse.

Although combining Machine Learning (ML) methodologies with real-time image data has been recognized to advance in wildfire science and management [51], there is an important restriction that a huge amount of training sets are required by ML techniques to learn the characteristics of the heat map of the burn area. Although computer vision is applied more and more on wildfire detection and measurement [52], few contributions can been found that provide a closed polygon for the parametrization of the wildfire perimeter on the basis of TIR or heat images. In [45], unsupervised edge detectors were applied to obtain the wildfire perimeter automatically, but performance in case of sparse TIR data had not been demonstrated.

The main contribution of the chapter is to provided a novel TIR image processing technique to characterize a closed polygon for the wildfire perimeter. To solve the problem of discontinuity of a heat image, the basic concept of Delaunay triangulation is used to obtain a convex polygon of a burn area. Similar ideas are explored in [23] where the so-called α-shape algorithm is used to determine the wildfire perimeter using information on hot spots. Unfortunately, the α-algorithm can only adjust the detected wildfire perimeter globally by changing the value of α, and is barely able to distinguish spot fires from the main burn area in an TIR image. To solve this problem, the novelty of the TIR image processing lies in the iterative trimming of triangular objects created by the Delaunay triangulation to obtain a closer match of the wildfire perimeter.

Both rough trimming and fine trimming are included in the iterative trimming method. For the rough trimming, after obtaining the convex polygon covering all pixels of the burn area by applying the Delaunay triangulation, two threshold values are created for this step. One is related to the longest side of the triangle created by Delaunay triangulation, and the other one is related to the relative burn area surrounding a vertex in a chosen domain. Based on these two threshold values, the iterative trimming method will first delete the redundant abnormally large triangle created by the vertex of the polygon, and then delete the isolated pixels of burn area caused by spot fire. As a result, a new convex polygon can be obtained relying on the remaining pixels of the burn area, and this process will be repeated iteratively until all the pixels of spot fires are removed. When the rough trimming is finished, another two threshold values related to the longest side of triangles connected to the vertex of the polygon and the relative burn area surrounding a vertex are created for the fine trimming. The wildfire perimeter can be finally obtained by tuning the threshold values. The performance of the iterative trimming method is illustrated by comparing the wildfire perimeter created by the iterative trimming method to those created by some classical edge detection methods, such as Canny edge detection, graph-cut method, and level set method.

4.2 Thermal Infrared Image of a Wildfire

One typical discontinuous RGB TIR image of wildfire is presented in Figure 4.1, and the corresponding R-value TIR image is presented in Figure 4.2. It can be observed from Figure 4.1 and Figure 4.2 that the bright area alternates with the dark area. Although the different bright areas are close, they are not connected. Delaunay triangulation is applied in this chapter to link up those bright areas.

To prepare for the Delaunay triangulation, the active pixels and inactive pixels are defined as

$$y_{i,j} = f([i,j],b) = \begin{cases} 0, & \text{if } b < b_t, \\ 1, & \text{if } b \geq b_t, \end{cases} \tag{4.1}$$

where i, j describe the location of the pixel, b is the R-value of the TIR image, and b_t is the threshold value to distinguish between active pixel and inactive pixel. $y_{i,j} = 1$ means the pixel at i, j is identified as an active pixel. It is part of burn area and can be used to detect the wildfire perimeter; $y_{i,j} = 0$ means the pixel at i, j is considered to be an inactive pixel. It represents either inactive wildfire or unburned area and should not be used for the detection of wildfire perimeter.

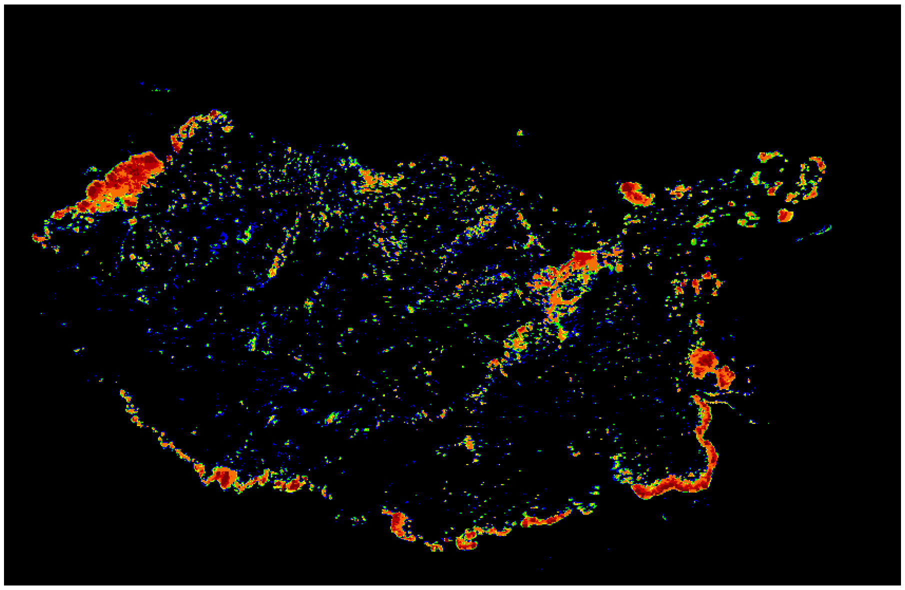


Figure 4.1. Example of an RGB TIR image, courtesy of DigitalGlobe WorldView-3 satellite data of the Happy Camp Complex fire. The size of the image is 850 $\times$ 550 pixels.

In this chapter, b_t is chosen to be 50. The reason is that the pixels with the R-value smaller than 50 belong to nearly completely dark area as shown in Figure 4.2.

4.3 Delaunay Triangulation and Iterative Trimming

4.3.1 Delaunay Triangulation

In 1934, Boris Delaunay introduces the Delaunay triangulation [53] that is well known for maximizing the minimum of all the angles of the resulting triangles. The Delaunay triangulation has been used in many applications due to the property of providing connectivity information for a given set of points [54]. This property is also adopted in this chapter to solve the problem of missing connection between the burn area.

By applying the Delaunay triangulation on the active pixels determined from Figure 4.2 in Section 4.2, the connecting triangles (red) can be established as shown in Figure 4.3(a). The union set of all the resulting triangles created by Delaunay triangulation is a convex hull

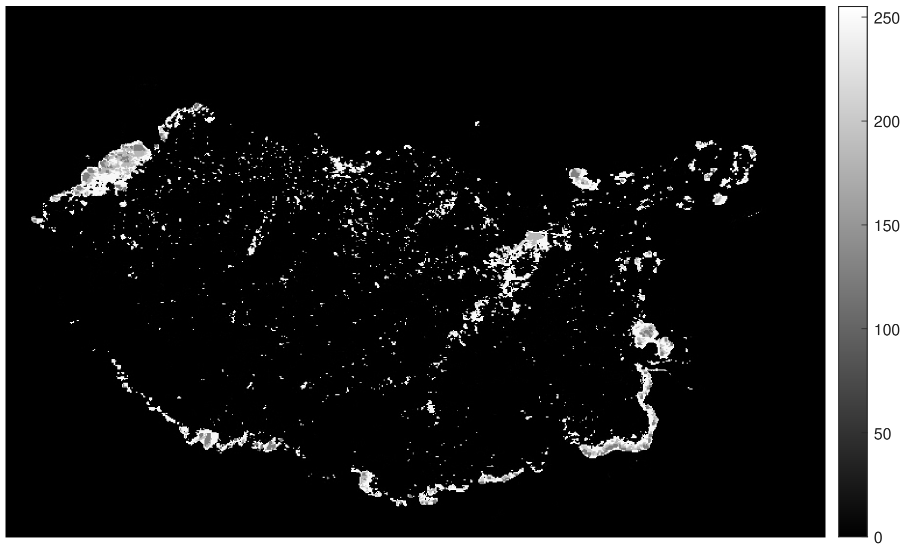
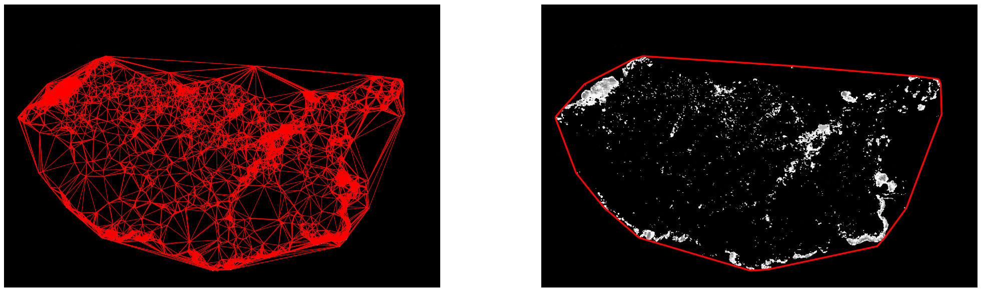


Figure 4.2. R-value of TIR image depicted in Figure 4.1.

depicted in Figure 4.3(b). It can be observed that the convex hull contains many redundant (sliver) triangles that hide the shape of the wildfire perimeter. These redundant large triangles are mostly caused by the vertices of the polygon characterizing the wildfire perimeter. Delaunay Triangulation connects these vertices as it did for the internal disconnected burn area. In addition, some active pixels are caused by spot fires and should also be removed to discover the main perimeter of the main wildfire. Therefore, it is important to distinguish the triangles outside the main wildfire perimeter from those inside the wildfire perimeter. To this end, an iterative trimming method is set up to trim the convex hull and reveal the wildfire perimeter.

4.3.2 Iterative Trimming Method

To cut out the redundant triangles established by Delaunay triangulation and the active pixels caused by spot fires, two iterative trimming steps are used: rough trimming and fine trimming. In the process of rough trimming, the abnormally large triangles and the pixels of spot fire will be removed iteratively. After that, the remaining polygon is further trimmed in the fine



(a) Delaunay triangulation on active pixels of Figure 4.2. (b) Polygon (red) by the union set of the triangles

Figure 4.3. Delaunay triangulation and convex hull.

trimming process. Details on both trimming processes are as follows.

Rough Trimming

As mentioned, the goal of rough trimming is to remove the abnormally large triangles created by Delaunay triangulation and some small groups of active pixels caused by spot fire to obtain a coarse shape of the main fire. The abnormally large triangles are selected by using the histogram of the longest side of all triangles. If the longest side of a triangle appears just once in a bin of the histogram and larger than the upper boundary of the last bin with count larger than two, then the triangle is categorized into the abnormally large triangle. The detailed procedure to recognize the abnormally large triangle is summarized in Algorithm 5.

Algorithm 5. Identifying abnormally large triangle

```
Input: Longest side of each triangle, $\mathbf{s}$
Output: Abnormally large triangle
1: Construct a histogram about the longest side of each triangles, $h$=histogram($\mathbf{s}$).
2: Find the index of the last triangle with count larger than one,
   $k$=find($h$.count $> 1$, ‘last’).
3: Abnormally large triangles are identified as the triangles with longest side larger than
   $l_r = k \times h$.binwidth.
```

Figure 4.4 shows the histogram calculated on the basis of the triangles depicted earlier in Figure 4.3(a). From the histogram in Figure 4.4, it can be observed that most of the triangles have a longest side smaller than 50, and abnormally large triangles can be recognized by choosing the last few triangles with one count in the histogram.

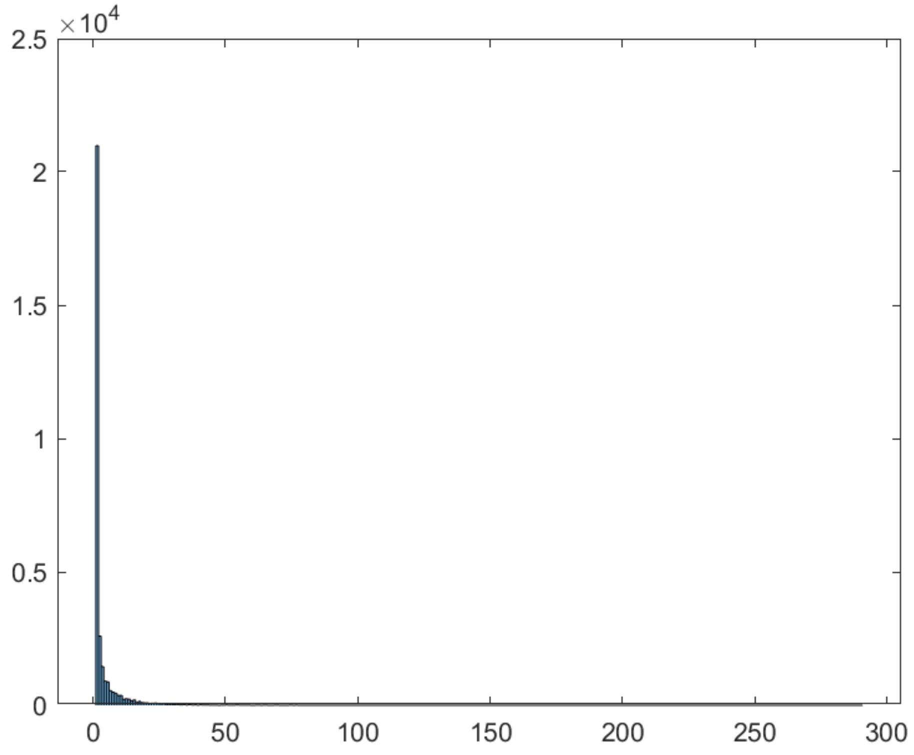


Figure 4.4. Histogram of the longest side of the triangles in Figure 4.3(a). The unit of the side length is (one) pixel length.

Due to the fact that spot fire is outside the main body of the wildfire, it is only necessary to check whether the vertices of the currently established polygon belong to the main fire or a spot fire. Spot fires are defined as a tiny connected burn area that is isolated from the main fire. Therefore, the active pixels of spot fires can be distinguished by the relative surface area within a chosen domain. The chosen domain can be a square patch of the TIR image centered by the vertex of the current polygon, and the relative surface area can be calculated by the ratio of the number of the connected active pixels containing the vertex of the current polygon and the number of pixels inside the whole square patch. The approach to select the active pixels of spot fires is summarized in Algorithm 6.

Algorithm 6. Identifying active pixels of spot fire

```
Input: Vertices of currently established polygon, $n$; size of square image patch, $m \times m$; threshold value, $d_u$.
Output: Active pixels of spot fires.
1: For each vertex, calculate the summation, $p_q,\ q = 1, 2, \dots, n$, of the number of all connected active pixels starting from the vertex inside the $m \times m$ square patch.
2: Compute the ratio $\frac{p_q}{m \times m}$, for $q = 1, 2, \dots, n$
3: If $\frac{p_q}{m \times m} \leq d_u$, all the connected active pixels with respect to the $q_{th}$ vertex of the polygon are regarded as part of the active pixels of spot fires.
```

As a consequence of removing the active pixels of spot fires, the set of points for Delaunay triangulation are affected. Hence, an iterative process of Delaunay triangulation, removing the abnormally large triangles, and removing the active pixels of spot fires is operated until all the active pixels of spot fires are eliminated. During the iteration, the number n of the vertices of the polygon will also change accordingly. After the rough trimming, a coarse shape of the main wildfire can be acquired, and the fine trimming will further remove the redundant triangles produced by Delaunay triangulation.

Fine Trimming

Instead of just removing the abnormally large triangles coarsely as done in rough trimming, fine trimming is more rigorous. With the goal of obtaining the perimeter of the main wildfire, the fine trimming is focused on the triangles connected with the vertices of the current polygon to avoid making the polygon disconnected. The redundant triangles can also be determined by the histogram of the longest side, or specific requirement on the wildfire perimeter.

Furthermore, considering the fact that the TIR image is sparse, the density of the active pixels around a vertex should also be taken into account. Heavier trimming can be done on the triangles connected to a vertex having a denser neighboring active pixels, and those triangles connected to a vertex that has a relatively sparse neighboring active pixels should be discarded more carefully. As a result, two threshold values are established for the fine trimming. The first one is the threshold value l_f, set for the longest side of the triangles that should be removed, and the second one is the threshold value d_l, set for the density of the neighboring active pixels around a vertex. The value of d_l is chosen so that the triangles connected to a vertex with a sparse neighboring active pixels will be protected from being removed even when the longest sides of the triangles are longer than l_f.

The density of the active pixels around a vertex can be measured similarly by the relative surface area $\frac{p_q}{m \times m}$ used in the Algorithm 6, where p_q is the number of all the connected pixels starting from the q_{th} vertex of the current polygon, and $m \times m$ is the size of the chosen square patch. All the triangles with a longest side larger than l_f and connected with a vertex that meet the requirement $\frac{p_q}{m \times m} > d_l$ will be removed in the fine trimming. During the process of trimming, new vertex will appear as the triangles are eliminated. Therefore, the trimming process should be operated iteratively until the number of the vertices of the polygon stays the same.

One of the main differences between the rough trimming and the fine trimming is that no active pixels are removed in the fine trimming. For this reason, the value of d_l can be less than the value of d_u. In other words, new vertex with $\frac{p_q}{m \times m} \leq d_u$ may appear during the fine trimming, and d_l can be chosen as $d_l < \frac{p_q}{m \times m} \leq d_u$ to trim the triangles connected to this newly created vertex. The complete procedure of the iterative trimming method including the step of Delaunay triangulation is summarized in Algorithm 7.

Algorithm 7. Iterative trimming method

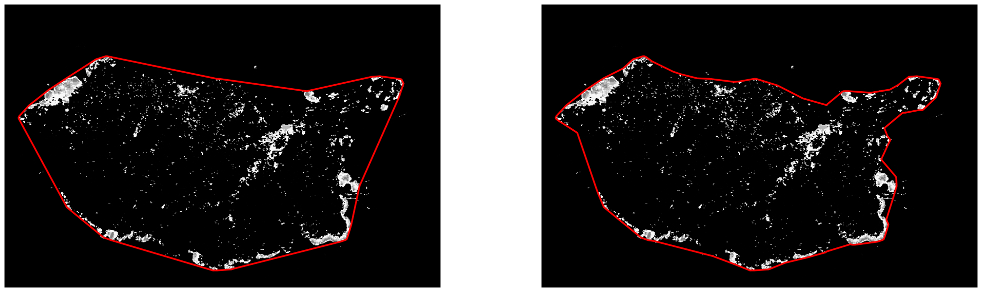
```
Input: TIR image of wildfire.
Output: Wildfire perimeter.
1: Determine the set of active pixels.
2: Find the locations of the active pixels, and apply the Delaunay triangulation on the active
   pixels.
3: Remove the abnormally large triangles and the active pixels caused by spot fires summarized
   in Algorithm 5 and Algorithm 6 respectively.
4: If the number of active pixels changes, go back to step 2.
5: Remove the triangle connected to the $q_{th}$ vertex with $\frac{p_q}{m \times m} > d_l$, if the longest side of this
   triangle is larger than $l_f$.
6: If the number of vertices of the polygon changes in step 5, repeat step 5.
```

The performance of the iterative trimming method based on the TIR data given earlier in Figure 4.1 is illustrated in the next section. In addition, the established polygon of the main wildfire perimeter obtained by the iterative trimming method on the basis of the Delaunay triangulation is compared to those obtained by the Canny edge detector, the graph-cut method, and the level set method.

4.4 Results and Discussion

4.4.1 Iterative Trimming Method

Considering the distribution of active pixels in Figure 4.2, the computed longest side of the triangles in Figure 4.3(a) and the resulting histogram depicted in Figure 4.4, the values of l_f, m, d_u, d_l are chosen as $50, 11, 0.16, 0.08$ respectively. The polygons obtained after the rough trimming and the fine trimming are shown in Figure 4.5(a) and Figure 4.5(b). It is worthwhile to note that, although the high-value infrared pixels are sparse and disconnected, the iterative trimming method can obtain a closed polygon of the wildfire automatically. It can also be noticed that some isolated active pixels outside the red polygon are regarded as the spot fires, and are not used to establish the polygon of the main wildfire. The computation time is around one second. As reference for the computation time, all calculations were performed on an Intel Core i7-7500U CPU with 16 GB RAM.



(a) Polygon (red) obtained by rough trimming. (b) Polygon (red) obtained by fine trimming.

Figure 4.5. Results of iterative trimming method.

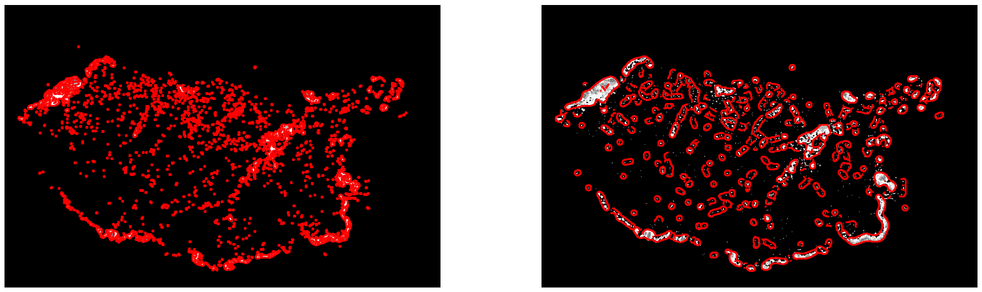
4.4.2 Canny Edge Detector

The Canny edge detector is developed by John F. Canny in 1986 [55]. Although Canny edge detection is a traditional edge detection method, it has been widely applied and improved in more recent researches [56–58]. It was also utilized in the study of wildfire monitoring [45] and was shown to be one of the most effective unsupervised detection algorithms. Three performance criteria: good detection, good localization, and unique response to a single edge, form the basis of Canny edge detector.

A Canny edge detection has five steps including smoothing the image by Gaussian filter, calculating the gradient of the image, deleting spurious response to true edges, using double threshold to find out prospective edge, and tracking edge by preserving strong edges and weak edges that are connected to strong edges. The image processing toolbox of MATLAB provides the standard edge detection algorithm, and Canny edge detector can be applied by calling the function `edge(I,'canny',threshold,sigma)`, where `I` is the image, `threshold` is used to ignore the unnecessary edges, and `sigma` decides the standard deviation of the Gaussian filter.

It is clear from the procedure of Canny edge detector that the image gradient is the basis for this edge detection method. For the TIR image given earlier in Figure 4.2, the Canny edge detector is more likely to work on detecting the boundaries of all isolated clusters of pixels instead of the main wildfire. To blur the image to a greater extent so that the effect of the isolation is reduced, a larger standard deviation of the Gaussian filter can be applied.

The results of Canny edge detection with default value for the `threshold` and two different standard deviations of the Gaussian filter are shown in Figure 4.6. Although increasing the standard deviation of Gaussian filter leads to a slightly better result, the Canny edge detector is still not able to detect the main boundary of the main wildfire.



(a) `sigma` = 1. (b) `sigma` = 5.

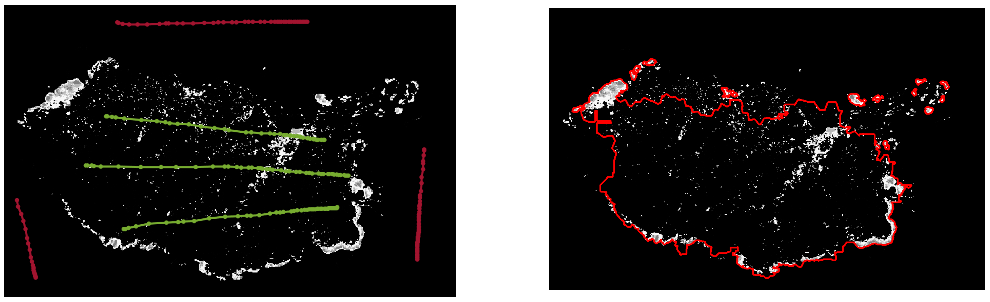
Figure 4.6. Boundaries (red) of the wildfire generated by Canny edge detector.

4.4.3 Graph-Cut Method

The graph-cut method is a widely used method in the field of computer vision and details can, for example, be found in [59–61]. Here a short summary of the graph-cut method is given. An image is first transformed to a graph consisting of nodes and edges, where edges are used to connect every two neighbor nodes, and nodes are composed of two terminal nodes, source node and sink node, and all pixels. Each edge is assigned with a weight or cost, and the goal of the graph-cut method is to find a minimum cut of the graph by using the max-flow min-cut theorem [62]. With the minimum cut, the image is divided into a foreground and a background.

The graph-cut method has a good ability to produce an optimal solution to the image segmentation of a binary problem, which is similar to distinguishing between the burn area and unburned area in a TIR image of a wildfire. Therefore, the graph-cut method is applied on Figure 4.2 to obtain the boundary of the wildfire via the image segmenter application in MATLAB. The seeds required by graph-cut method are highlighted as Figure 4.7(a), and the result of the graph-cut method is shown in Figure 4.7(b).

It can be observed that graph-cut method only works well on identifying part of the wildfire boundary near the provided seeds. Although a good wildfire perimeter can be created by graph-cut method when more detailed and complicated seeds are provided, too much human interaction is required and the work involved is almost the same as a manual delineation, limiting



(a) Seeds for foreground (green) and background (red). (b) Boundaries (red) generated by graph-cut methods.

Figure 4.7. Results of graph-cut method.

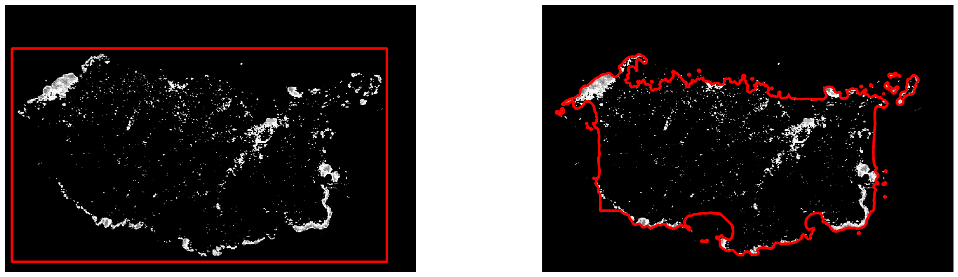
the application in automatic wildfire perimeter detection.

4.4.4 Level Set Method

The level set method is an impressive tool for image segmentation by exploiting the information of regions and boundaries of the object [63]. It applies level sets for numerical analysis of surfaces, and the application of level sets makes it beneficial to track the change of the topology, such as the development of a hole. In addition, the level set method provides an implicit description of the object without the need of parameterizing the object. Due to the fact that a level set method has a good ability to separate two regions, it might be a solution to the binary problem of detecting the burn area from the unburned area.

Unfortunately, for a disconnected TIR image of a wildfire as depicted in Figure 4.2, it is infeasible to decide whether a hole exists, or what the size of the hole is. Moreover, continuous image gradient and intensity of the pixels are important components of level set method to determine the speed of the evolving and the shape described by each level set. Another potential problem of the level set method is that the final result is dependent on the choice of the initial contour, but the shape of the wildfire can be arbitrary. Hence, an algorithm of automatically generating an initial zero-level contour is required, or a fire expert needs to provide an initial contour for each image of a wildfire to apply the level set method.

To test the performance of the level set method on Figure 4.2, the level set method introduced by [64] is adopted. By choosing the parameters $\alpha = 2$, $\lambda = 5$, $\mu = 0.01$, the result can be achieved after 200 iterations as Figure 4.8(b), with the initial contour established as Figure 4.8(a). It can be observed that the boundary captured by level set method passes through the disconnected part of the burn area of the TIR image. The reason for that is level set method also relies on the image gradient or pixel intensity to calculate the level sets, and there is a huge change in the image gradient between the disconnected areas.



(a) Initial zero level contour.

(b) Final zero level contour

Figure 4.8. Results of level set method.

4.5 Summary

This chapter introduces an iterative trimming method (ITM) based on Delaunay triangulation with a goal to establish a closed polygon of the main wildfire perimeter automatically for a TIR image of wildfire. Although the burn area caused by spot fire is deleted in the ITM, they can be captured respectively by treating each burn area of a spot fire as a main wildfire perimeter. The performance of the iterative trimming method is validated by providing a study that compares the result of the ITM with those of the various edge detection methods based on a satellite generated TIR image. The comparison study shows that the various edge detection methods fail to provide a single closed polygon that parametrizes the main wildfire perimeter. Often, disconnected burn areas will be detected separately. The proposed ITM shows good performance with a single closed polygon for the wildfire perimeter. For further studies, more information of the wildfire can be used in the iterative trimming method. For example, the wind direction and wind speed can be used to predict the location of the spot fire, and a priori knowledge of the spot fire can be included in the iterative trimming method to better capture and remove the active pixels caused by spot fires.

4.6 Acknowledgement

Chapter 4, in full, has been accepted for publication of the material as it may appear in "Wildfire Perimeter Detection via Iterative Trimming Method", Tan, L., Hu, Y., Tan, S., de Callafon, R. A., & Altıntaş, I. (2023, July). In Computational Science–ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023. The dissertation author was the primary investigator and author of this paper.

Chapter 5

Ensemble Based Learning for Automated Safety Labeling of Prescribed Fires

Prescribed fires are controlled burns of vegetation that follow a burn plan to reduce fuel build-up and mitigate unanticipated wildfire impacts. To understand the risks associated to a prescribed burn, modern fire simulation tools can be used to simulate the progression of a prescribed fire as a function of burn conditions that include ignition patterns, wind conditions, fuel moisture and terrain information. Although fire simulation tools help characterize fire behavior, the unknown non-linear interactions between burn conditions requires the need to run multiple fire simulations (ensembles) to formulate an allowable range on burn conditions for a burn plan. Processing the ensembles is often a labor intensive process run by user-domain experts that interpret the simulation results and carefully label the safety of the prescribed fire. The contribution of this chapter is an algorithm of ensemble based learning that automates the safety labeling of ensembles created by a modern fire simulation tool. The automated safety labeling in this algorithm is done by first extracting important prescribed fire performance metrics from the ensembles and learn the allowable range of these metrics from a subset of manually labeled ensembles via a gradient free optimization. Subsequently, remaining ensembles can be labeled automatically based on the learned threshold values. The process of learning and automatic safety labeling is illustrated on 900 ensembles created by QUIC-Fire of a prescribed fire in the Yosemite, CA region. The results show a performance of over 80% matching of learned automated safety labels in comparison to manually generated safety labels created by fire domain experts.

5.1 Introduction

As the extent of landscapes burned by wildfires continuously grow, it is important to take advantage of prescribed fires to manage the risk of uncontrollable wildfires. A prescribed fire is a controlled burn of vegetation and ignited intentionally to meet fuel and vegetation management objectives, such as reducing hazardous fuels, sustaining the natural landscapes, and avoiding extreme wildfires. Compared to a wildfire that is unplanned, prescribed fire can be controlled by reducing the risk of a fire escape.

There are many positive effects of a prescribed fire on soil, vegetation, or even some cultural artifacts, and periodic fire plays an important role in the balance of many ecosystems [65–69]. Therefore, prescribed fire can be used as a tool to manage the forest area in various ecological aspects, such as preventing invasive vegetation and facilitating the recovery of specific species [15]. However, people are averse to the risk of a prescribed fire due to the lack of scientific knowledge about the benefit of a prescribed fire in an ecosystem management [70].

Environmental or burn conditions that include the landscape, terrain, fuel moisture, wind speed, wind direction and ignition pattern are important factors for the progression of the fire. Extensive modeling efforts have been documented that help with the prediction of the fire spread as a function of the burn conditions [22, 40, 71–74]. With the advance of the science and technology, various software tools have been developed to numerically simulate the progression of a prescribed fire. QUIC-Fire [13] is a three-dimensional fire simulation tool that provides dynamic fuel consumption over time.

Although the progression of the consumed fuel can be simulated by QUIC-Fire, the trade-off between controlled fuel consumption and the safety of the prescribed burn must be taken into account when deciding on the allowable burn conditions. In practice, the unknown non-linear interactions between burn conditions requires the need to run multiple QUIC-Fire simulations (ensembles). The ensembles can be labeled as safe, marginal and unsafe by fire domain experts manually to formulate an allowable range on burn conditions. The manual labeling process is labor intensive and time consuming, and a fast and accurate automatic labeling algorithm that incorporates and learns the expertise of a fire domain expert is desirable.

The contribution of this chapter is an algorithm of ensemble based learning that automates the safety labeling of ensembles created by a modern fire simulation tool. The automated safety labeling in this algorithm is done by first extracting important prescribed fire performance metrics from the ensembles based on a desired burn boundary within a burn plan. Any fire escapes outside the desired burn boundary is characterized as a *slop-over* and performance metrics identify the size, spacing and the number of slop-overs. Subsequently, manually labeled ensembles are used to learn the allowable range of the slop-over metrics to distinguish between safe, marginal and unsafe fire conditions. With some integer-valued metrics, the learning is formulated via an gradient-free optimization based on a genetic algorithm [75] that has the capability to deal with integer-valued functions.

The optimized (learned) allowable range of the slop-over metrics and the environmental conditions such as wind speed and fuel moisture are configured as parameters in the automatic labeling. The numerical values of these parameters are used in the automatic labeling algorithm. The optimization ensures an optimized prediction accuracy of the automatic safety labeling of the ensembles. In order to authenticate the performance of the automatic labeling, the use cases of 900 ensembles of a prescribed fire in the Yosemite, CA area are utilized. Learning and matching 100% of the manually assigned safety labels of a subset with 48 out of the 900 ensembles, the automatic labeling is used to provide safety labels for the remaining ensembles. With a success rate above 80%, the proposed automatic labeling algorithm works efficiently and accurately, and can be used as a tool to design the burn plan of the prescribed fire.

5.2 QUIC-Fire Output Data

With the information of the surface moisture, fuel type, wind conditions, and ignition pattern, QUIC-Fire [13] can simulate the spread of the prescribed fire. The typical output produced by QUIC-Fire at each simulation step is the fuel consumption as depicted in Figure 5.1.

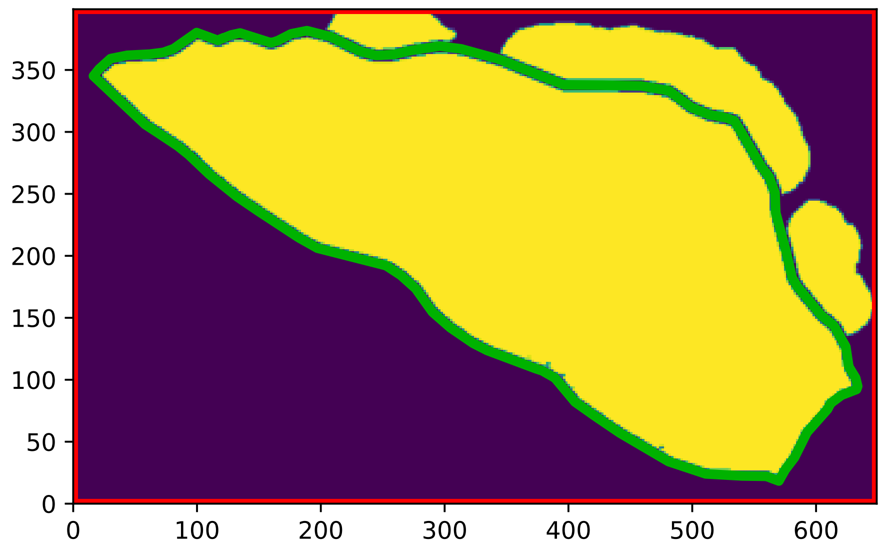


Figure 5.1. Output (fuel consumption) of QUIC-Fire. Desired boundary (green) and allowable boundary (red). Burn area (fuel consumed) with $y = 1$ (yellow) and unburned area (fuel not consumed) with $y = 0$ (dark blue).

Similar to [76], the burn area is represented by the yellow area with $y = 1$, and the unburned area is represented by dark blue area with $y = 0$. The value of each pixel can be expressed as

$$y_{i,j} = f([i,j],b) = \begin{cases} 0, & \text{if } b < 0.001 \\ 1, & \text{if } b \geq 0.001 \end{cases} \tag{5.1}$$

where $[i, j]$ describes the position of the target pixel in the image, b is the absolute difference value between the fuel densities before the prescribed fire starts and after the prescribed fire ends, and y is the value of pixel at $[i, j]$ and used to distinguish between the burn area and unburned area.

In Figure 5.1, the provided desired boundary is drawn by a green line, which defines the area inside the desired boundary that is expected to burn, and the allowable boundary is drawn by red line, which separates buffer area and non-allowable burn area where fire is definitely considered to be unsafe. The allowable boundary is determined by the size of the fuel domain used for the simulation in QUIC-Fire. Without loss of generality, whether a fire escapes outside the allowable boundary can be distinguished by checking whether there is a pixel with $y = 1$ outside the closed polygon representing the allowable boundary. On account of the fact that the shape of the fire is arbitrary, deciding the fire safety by only depending on predetermined boundaries is not enough.

The yellow area outside the desired boundary in Figure 5.1 is regarded as the slop-over, and the number of the disconnected yellow area outside the desired boundary is regarded as the number of slop-overs. Hence, the simulation shown as Figure 5.1 includes three slop-overs. With the definitions of desired boundary and allowable boundary, slop-over plays an important role in evaluating the safety of the prescribed fire. If the slop-over has the potential to spread outside the allowable boundary, and is hard to control, the corresponding prescribed fire can be unsafe. For identification of the fire safety for each simulation (ensemble), three levels are used: safe, marginal and unsafe.

5.3 Feature Definitions

Following the summary of the nomenclature given in Table 5.1, a short explanation is given for the inputs and parameters used in the automatic labeling of ensembles. After collecting the manual labels provided by fire domain experts, the number of the slop-over k_s, the total area of the slop-over A_s, and the distance between each slop-over l_s, can be used to evaluate the safety of the prescribed fire. The total area of the slop-over directly reflects the result of a simulated prescribed fire. Hence, it is an important factor in measuring the fire safety. Limited by the

Table 5.1. Nomenclature of inputs and parameters for automatic labeling

Inputs:	
A_s	total area of slop-overs
k_s	number of slop-overs
l_s	distance between each slop-over
w_s	wind speed
s_s	surface moisture
Parameters:	
A_{max}	maximum allowed total area of slop-overs
A_{mar}	marginally allowed total area of slop-overs
k_{max}	maximum allowed number of slop-overs
α	expansion coefficient
β	expansion coefficient
l_{max}	maximum allowed distance between each slop-over
l_{mar}	marginally allowed distance between each slop-over
w_t	threshold value of wind speed
s_t	threshold value of surface moisture
k_{t_1}	the first threshold value of number of slop-over
k_{t_2}	the second threshold value of number of slop-over

number of firefighters, large number of slop-over or large distance between slop-overs can both result in an uncontrollable prescribed fire.

To quantitatively measure these three terms, some parameters are created in the automatic labeling algorithm. A_{max} and A_{mar} represent the maximum and marginally allowable total area of slop-over, k_{max} denotes the maximum allowable number of slop-over, and l_{max} and l_{mar} indicate the maximum and marginally allowable distance between each slop-over. In addition to simply exploiting the information of slop-over, the complex environmental conditions are also taken into consideration.

For additional flexibility, the parameters α and β are utilized as the constant amplification coefficients to enlarge the potential risk of the slop-over. As a prescribed fire can be more dangerous when the wind speed is higher, the surface moisture is lower, and the number of the slop-over is larger, four threshold values are created to better distinguish the effect of the total area of slop-over and the distance between each slop-over in different situations. The parameters k_{t_1} and k_{t_2} are established as the number of the slop-over when the risk level of prescribed fire varies significantly, while s_t and w_t are the threshold values for the surface moisture and wind speed respectively. If wind speed is larger than w_t, or surface moisture is smaller than s_t, more caution is required to decide the risk level of the prescribed fire. With these parameters, an automatic safety labeling algorithm can be formulated.

5.4 Automatic Labeling Algorithm

5.4.1 Postprocessing of QUIC-Fire Output

To calculate the previously mentioned A_s, k_s and l_s for each ensemble of prescribed burn, the slop-overs should be characterized by removing the burn area inside the desired boundary as shown in Figure 5.2. Following the definition of y in (5.1), all slop-overs have $y = 1$ as shown in Figure 5.2(a).

To further distinguish the slop-overs, different non-zero values for y are assigned to different slop-overs. For the numerical implementation, `label` function in the package of scikit-image [77] in Python is a good tool to achieve this goal. It first detects the slop-overs according to the connectivity, and then assigns different values of y to different slop-overs. From Figure 5.2(b) it can be observed that three slop-overs are plotted by different colors, where yellow, magenta, and cyan correspond to $y = 1$, $y = 2$, and $y = 3$ respectively.

Afterwards, the number of the slop-over can be determined by the number of different non-zero values of y, and the area of each slop-over can be calculated by summing up the number of pixels with corresponding y. Finally, the distances between the centers of the smallest vertically oriented rectangles that separately contain each slop-over serves as the distances between each slop-over.

5.4.2 Process of Automatic Labeling

Since a distance between slop-overs exists only when there are more than one slop-over, the number and the total area of the slop-over are more important and are utilized first for labeling

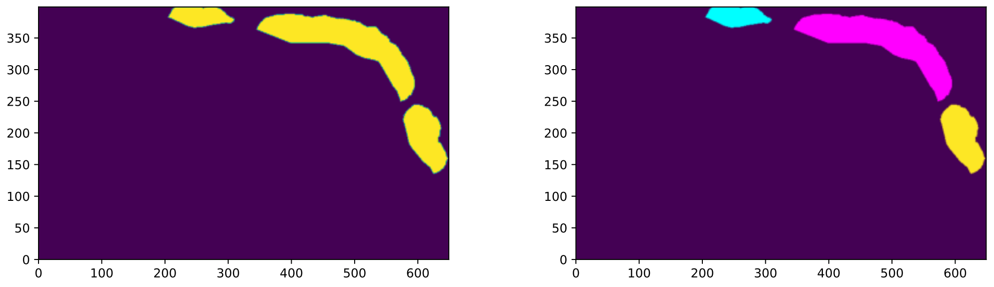

(a) Plot of slop-overs with same y. (b) Plot of slop-overs with different y.

Figure 5.2. Extracted slop-overs.

the fire safety. Due to the limited resource of the fire fighting, it is impossible to control the slop-overs of one prescribed fire simultaneously if multiple slop-overs are far away from each other. Therefore, the distance between slop-overs should also be measured.

Additionally, wind speed and surface moisture around the prescribed fire also affect the fire spread. Even a small slop-over can grow out of control in a short time when the wind speed is large and the surface moisture is low. To account for these situations, two expansion coefficients α and β are applied on the total area of the slop-overs to reflect the emphasis on the effect of the extreme environment. For each ensemble, with computed metrics A_s, k_s and l_s, and provided data of w_s and s_s, the automatic process can be described as follows.

At first, the prescribed fire ensemble is assumed to be safe. Any prescribed fire ensemble with the total area $A_s > A_{max}$, or number of slop-overs $k_s > k_{max}$ is labeled to be unsafe. When the wind speed $w_s > w_t$, the surface moisture $s_s < s_t$, and the number of the slop-over $k_{t_1} < k_s \leq k_{max}$, the prescribed fire is more likely to be unsafe. For that purpose, αA_s is compared to A_{max}. If $\alpha A_s > A_{max}$, the prescribed fire ensemble is regarded as an unsafe fire.

To evaluate the safety of a prescribed fire ensemble by the distance between slop-overs, a prescribed fire ensemble with the number of the slop-over $k_{t_2} < k_s \leq k_{max}$, and the maximum distance between each slop-over $max(l_s) > l_{max}$ is classified as unsafe. If a prescribed fire is not unsafe, then it will be checked whether it is marginal. The process of judging whether a prescribed fire is marginal is similar, and another expansion coefficient β is set up to put more cautions in the judgement when the environment is more suitable for the spread of the prescribed fire. The automatic labeling algorithm is summarized in Algorithm 8.

Algorithm 8. Automatic Labeling Algorithm

```
Inputs: $A_s$, $k_s$, $l_s$, $w_s$, $s_s$
Parameters: $A_{max}$, $A_{mar}$, $k_{max}$, $\alpha$, $\beta$, $l_{max}$, $l_{mar}$, $w_t$, $s_t$, $k_{t_1}$, $k_{t_2}$
Output: Label of the simulated prescribed fire
 1: Assume the prescribed fire is safe at the beginning.
 2: if the prescribed fire move outside the allowable boundary then
 3:     the prescribed fire is unsafe
 4: else if $A_s > A_{max}$ then
 5:     the prescribed fire is unsafe
 6: else if $k_s > k_{max}$ then
 7:     the prescribed fire is unsafe
 8: else if $k_{t_1} < k_s \leq k_{max}$ and $w_s > w_t$ and $s_s < s_t$ and $\alpha A_s > A_{max}$ then
 9:     the prescribed fire is unsafe
10: else if $k_{t_2} < k_s \leq k_{max}$ and $max(l_s) > l_{max}$ then
11:     the prescribed fire is unsafe
12: else if $A_s > A_{mar}$ then
13:     the prescribed fire is marginal
14: else if $k_{t_1} < k_s \leq k_{max}$ and $w_s > w_t$ and $s_s < s_t$ and $\beta A_s > A_{mar}$ then
15:     the prescribed fire is marginal
16: else if $k_{t_2} < k_s \leq k_{max}$ and $max(l_s) > l_{mar}$ then
17:     the prescribed fire is marginal
18: end if
```

5.5 Optimization

It is clear that the accuracy of the automatic labeling is dependent on the numerical values of the parameters listed in Table 5.1. The numerical values of the parameters can be optimized by using safety labels created by fire domain experts. The formal problem of learning the numerical parameters on the basis of manually labeled fire safety data can be stated as the optimization

$$\begin{gathered}\min_{\mathbf{u}} \sum_{i=1}^{N} d_i \frac{u_i}{q_i} - c(\mathbf{u}), \\ \text{subject to: } p_i \leq u_i \leq q_i \text{ for } i = 1,2,\ldots,N \\ A_{mar} \leq A_{max},\ \ l_{mar} \leq l_{max},\ \ k_{t_1}, k_{t_2} \leq k_{max} \\ c(\mathbf{u}), A_{mar}, A_{max}, k_{t_1}, k_{t_2}, k_{max}, w_t \in \mathbb{Z}\end{gathered} \tag{5.2}$$

where $\mathbf{u} = [\alpha, \beta, A_{mar}, A_{max}, k_{max}, k_{t_1}, k_{t_2}, l_{mar}, l_{max}, s_t, w_t]$, N is the number of parameter, and d_i is the weighting coefficient with $\sum_{i=1}^{N} d_i = 1$. u_i represents the i_{th} parameter in $\mathbf{u}$, and p_i and q_i are the lower bound and upper bound of the i_{th} parameter respectively. The value of p_i and q_i can be obtained from the burn plan that includes the information of the fuel domain, the wind conditions and the surface moisture for the simulated prescribed fire.

In (5.2), $c(\mathbf{u})$ denotes the number of safety match between the automatic labels created by Algorithm 8 and the manual labels created by a fire domain expert, where $\mathbf{u}$ represents the parameters. With $u_i \leq q_i$ from Equation 5.2 and $\sum_{i=1}^{N} d_i = 1$, it can be verified that

$$\sum_{i=1}^{N} d_i \frac{u_i}{q_i} \leq \sum_{i=1}^{N} d_i = 1 \tag{5.3}$$

since $c(\mathbf{u})$ is the number of matches between the automatic labeling and manual labeling, any change in $c(\mathbf{u})$ when $\mathbf{u}$ varies is greater than or equal to one.

With (5.3), an inequality can be derived for the change in $c(\mathbf{u})$, denoted by $\Delta c(\mathbf{u})$, when varying $\mathbf{u}$. The value of $\Delta c(\mathbf{u})$ is bounded by

$$\sum_{i=1}^{N} d_i \frac{u_i}{q_i} \leq 1 \leq \Delta c(\mathbf{u}) \tag{5.4}$$

and therefore the optimization will first focus on increasing the number of matches between the automatic labels and manual labels, and then decrease the numerical value of the parameters. As a result, the parameters obtained by the optimization will achieve the goal of gaining the maximum match number with the necessary minimum values of the parameters, representing the allowable range on the slop-over metrics.

Because $c(\mathbf{u})$ is an integer-valued function, and there is no analytic expression of $c(\mathbf{u})$, a gradient-free optimization method that can also deal with the integer-valued function should be applied. The genetic algorithm is an explicit and effective solution to this problem. The genetic algorithm will repeatedly modify the population of individual solution. Three steps are included in the genetic algorithm. At first, a random initial population is created. Then, a sequence of new populations are created iteratively based on the previous populations by scoring the fitness of each member of the population, selecting pairs of the members relied on the fitness, and generating the new population by applying crossover and mutation. The last step is to stop the algorithm when the change in value of the fitness function for the best member is less than a tolerance value, or after a predetermined maximum number of iteration. The procedure of the genetic algorithm is summarized in Algorithm 9.

Algorithm 9. Genetic Algorithm

Input: Population size m, maximum number of iterations t_{max}, and stopping criterion ε

Output: Global optimal solution, $\mathbf{u}_{opt}$

```
1: Create the initial $m$ members $\mathbf{u}_j (j = 1, 2, \dots, m)$ of population, and let $t = 0$.
2: Scoring the fitness value of each member, and find the member with best fitness value $f(t)$.
3: select pairs of members from previous population based on fitness value.
4: Apply crossover and mutation to generate the new population.
5: $t = t + 1$.
6: Stop when $t = t_{max}$ or $f(t) - f(t-1) < \varepsilon$; Otherwise, go back to step 2 to repeatedly modify
   the population.
```

5.6 Numerical results

5.6.1 Ensemble Based Learning

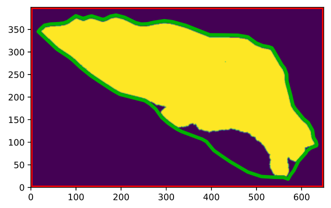
For illustration of the ensemble based learning for automated safety labeling, two fire domain experts work together to manually label the fire safety of 900 ensembles of a prescribed fire in the Yosemite, CA region. QUIC-Fire simulations for the 900 ensembles are created by varying ignition patterns, wind speed, wind direction and fuel moisture for each of the ensembles.

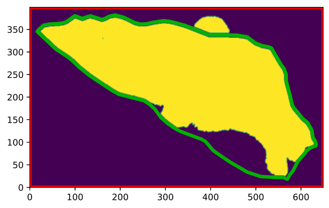
To ensure the validity of the manual labels used for learning, 48 out of the 900 ensembles are labeled by two fire domain experts separately and carefully. Some typical cases in these 48 ensembles with same manual labels are shown as Figure 5.3. In Figures 5.3(a), 5.3(b), 5.3(c) and 5.3(d), the total area of the slop-overs are small enough and there is a certain distance between the slop-overs and the allowable boundary. Hence, they are labeled as safe prescribed fires. It can be noticed that the sizes of the slop-overs in Figures 5.3(e) and 5.3(f) are relatively large, and the top parts of the slop-overs are fairly close to the allowable boundary. Therefore, the safeties of these two prescribed fire are labeled to be marginal. For Figure 5.3(g), the prescribed fire crosses the allowable boundary, and for Figure 5.3(h), the total area of the slop-overs is larger than A_{max} despite that the fire does not escape outside the allowable boundary. As a result, both of them are considered to be unsafe. At last, the slop-overs in Figure 5.3(i) are large and cross the allowable boundary. Therefore, these slop-overs obviously constitute unsafe prescribed fire conditions.

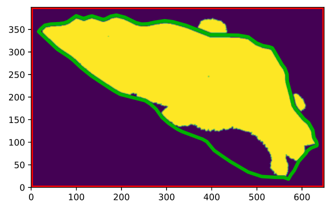
The ensemble based learning only uses the 48 out of the 900 ensembles, whereas the remaining 852 labels will be labeled automatically based on the optimized parameter values of Table 5.1 obtained by the learning. For cross validation of the accuracy of learning and labeling, both the 48 ensembles used for learning and the 852 ensembles not used for learning but used for labeling only are compared.

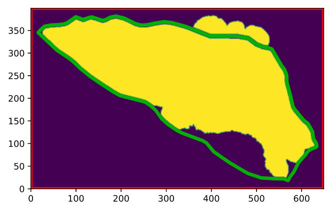
From this cross validation, a 100% accuracy has been achieved for both sets (48 ensembles) of the manual labels created by two fire domain experts respectively. As expected, the optimized parameters for each fire domain expert are slightly different, as each fire domain expert will interpret and label the ensembles slightly differently. As such, the rules used by the two fire domain experts are not completely the same, but either of them can be captured by the learning algorithm. The parameters, $\mathbf{u}^1$ and $\mathbf{u}^2$, optimized by the two sets of manual labels are summarized in Table 5.2.

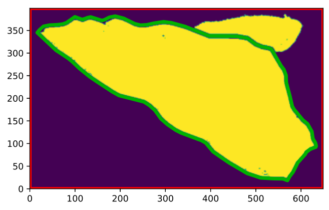
By inspecting the numerical values of the parameters in Table 5.2, it can be observed that fire domain expert, from whom $\mathbf{u}^1$ is learned, is more cautious than the fire domain expert, from

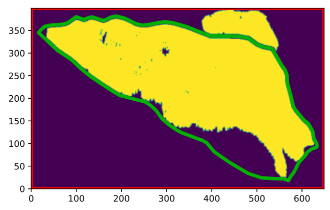

(a) safe

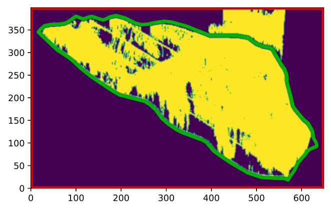

(b) safe

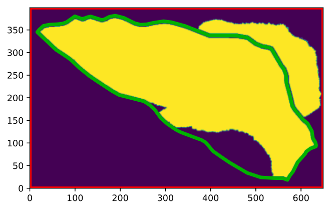

(c) safe


(d) safe


(e) marginal


(f) marginal


(g) unsafe


(h) unsafe

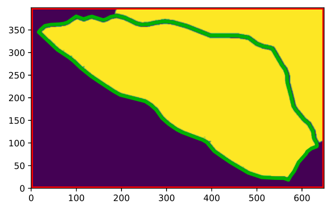

(i) unsafe

Figure 5.3. Typical cases in the training data set.

whom $\mathbf{u}^2$ is learned, because $\mathbf{u}^1$ has smaller values for A_{mar} and A_{max}. Furthermore, α in $\mathbf{u}^1$ is larger than one, and β in $\mathbf{u}^1$ is close to one. This means $\mathbf{u}^1$ is more focused on identifying unsafe prescribed fire conditions. In addition, $\mathbf{u}^1$ takes advantage of the distance between each slop-over to judge the risk level of the prescribed fire. In comparison, $\mathbf{u}^2$ has a higher tolerance for the threat of a prescribed fire, and $\mathbf{u}^2$ has higher probability to assess the risk level of prescribed fire as marginal instead of unsafe with α close to one and β larger than one. Since both l_{mar} and l_{max} in $\mathbf{u}^2$ are close to zero, it can be assumed that the distance between each slop-over is not utilized, which is also supported by $k_{t_2} = k_{max}$.

Table 5.2. Parameters, $\mathbf{u}^1$ and $\mathbf{u}^2$, optimized by the two sets of 48 manual labels respectively.

	A_{mar}	A_{max}	α	β	w_t	s_t	l_{mar}	l_{max}	k_{t_1}	k_{t_2}	k_{max}
$\mathbf{u}^1$	8280	10655	1.31	1.01	0	0.20	133.31	263.72	1	2	3
$\mathbf{u}^2$	9723	17719	1.04	1.37	3	0.35	1.06	1.96	0	3	3

5.6.2 Automated Safety Labeling

Due to the large workload for a single fire domain expert to label the remaining 852 ensembles, the 852 manual labels are created together by the two fire domain experts, effectively mixing their fire safety labeling expertise in the remaining data set of ensembles. To cross validate the performance of the automatic labeling, manual labels of the 852 ensembles, not used for learning, are compared to the labels created by Algorithm 8 with the parameters values $\mathbf{u}^1$ and $\mathbf{u}^2$ listed in Table 5.2 separately.

The match accuracy between the manual labels and the automatic labels created using $\mathbf{u}^1$ is 76.76%; the match accuracy between the manual labels and the automatic labels created using $\mathbf{u}^2$ is 76.88%; the match accuracy between the manual labels and the automatic labels created using either $\mathbf{u}^1$ or $\mathbf{u}^2$ is 80.52%. As a consequence, more than 80% manual labels can be captured by the automatic labeling using either $\mathbf{u}^1$ or $\mathbf{u}^2$.

5.6.3 Re-evaluation of Manual Labeling

To investigate the inconsistency between manual and automatic labeling, 12 ensembles (Figure 5.4) are chosen as canonical cases from the 852 ensembles, in which the manual labels are different from the automatic labels created using either $\mathbf{u}^1$ or $\mathbf{u}^2$. Without loss of generality and with the purpose of reducing the workload, only one fire expert, by whose manual labels $\mathbf{u}^1$ was optimized, relabeled these 12 ensembles and 10 revised manual labels were the same as the automatic labels created by Algorithm 8 with $\mathbf{u}^1$.

In Figures 5.4(a) and 5.4(b), the total area of the slop-overs is small enough. Therefore, both of them should be regarded as safe prescribed fires. In addition, Figure 5.4(b) is similar to Figure 5.3(c), which further confirms that the fire shown in Figure 5.4(b) is safe. For

Figures 5.4(c) and 5.4(d), since the slop-overs are larger and hard to control, they should be unsafe. For Figures 5.4(e) to 5.4(i), the prescribed fires cross the allowable boundary in different locations and are unsafe. It is worthwhile to note that the re-evaluation helped to further improve the number of match between manually and automatically created labels by correcting the previous manually applied safety labels.

Since there is no ensemble with four or more slop-overs included in the 48 ensembles used for learning and more slop-overs will lead to more dangerous fire conditions, k_{max} is optimized as 3, and all the prescribed fires with four or more slop-overs will be considered as unsafe fires in the automatic labeling. For Figures 5.4(j) and 5.4(k), both of them have four slop-overs and the difference between them is that all four slop-overs in Figure 5.4(j) stay together while one slop-over is far away from the other three slop-overs in Figure 5.4(k), which further increase the difficulty in controlling the prescribed fire. Hence, the prescribed fires shown by Figure 5.4(j) and Figure 5.4(k) are considered to be marginal and unsafe respectively by the fire expert.

5.6.4 Further Improvements

To further improve the automatic labeling, a user-defined marginally allowed number of slop-overs k_{mar} and a user-defined maximum allowed number of slop-overs k_{max} can be imported into Algorithm 8. Expanding the training data to include more scenarios can also improve the performance of automatic labeling at the price of having to provide more manually labeled ensembles.

To make sure the user-defined k_{mar} and k_{max} will not change during the optimization process, two more linear equality constraints on k_{mar} and k_{max} can be added to (5.2). At last, for Figure 5.4(l), even the fire domain expert cannot give an exact answer based on the current data. It means more information, like topography, vegetation, and contingency resources, is needed.

In summary, the automatic labeling, Algorithm 8, has a good ability to create the label for the safety of the prescribed fire. Since the label is created by measuring the number of slop-overs, the total area of the slop-over, and the distance between each slop-over, the automatic labeling can not only create the label but also give a feedback about which rule is used to create the label so that people can get access to the interpretation of the automatic labeling.

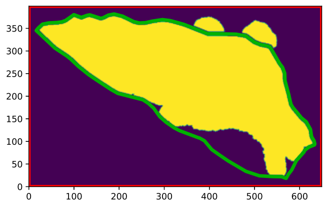
5.7 Summary

This chapter introduces an automatic labeling algorithm to establish the safety label for each ensemble of a simulated prescribed fire. The automatic labeling is based on prescribed fire safety metrics that include the number of slop-overs, the total surface area of slop-overs, and the distance between slop-overs. In addition to the safety label, the automatic labeling algorithm can provide an explanation why a prescribed fire is considered to be safe, marginal, or unsafe. Necessary parameters are optimized in the automatic labeling algorithm via a genetic algorithm to assist in determining the label of each ensemble of the simulated prescribed fire. A numerical validation based on 900 ensembles with manually generated safety labels of a prescribed fire in the Yosemite, CA area showed a 100% match of safety labels for the training data (48 out of 900 ensembles) and a larger than 80% match on the cross validation of safety labels not used in the training data (852 out of 900 ensembles).

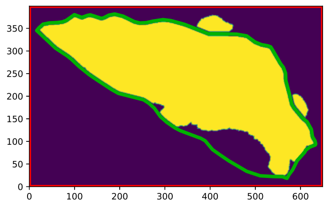
5.8 Acknowledgement

We thank Matthew Snider and J. Kevin Hiers for providing the manual safety labels of the fire simulations.

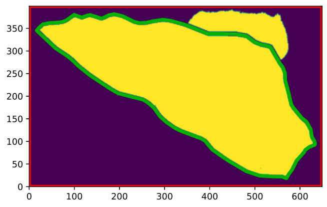
Chapter 5, in full, has been accepted for publication of the material as it may appear in "Ensemble Based Learning for Automated Safety Labeling of Prescribed Fires", Tan, L., de Callafon, R. A., Nguyen, M., & Altıntaş, I. (2023, July). In Computational Science–ICCS 2023: 23rd International Conference, Prague, Czech Republic, July 3–5, 2023. The dissertation author was the primary investigator and author of this paper.



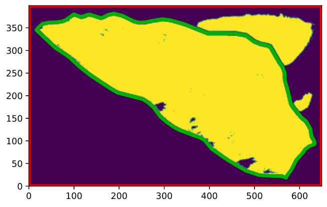
(a) PML: unsafe
AL: safe
RML: safe



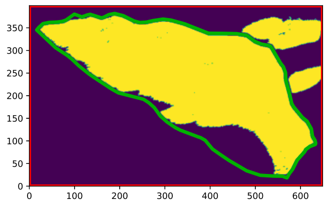
(b) PML: unsafe
AL: safe
RML: safe



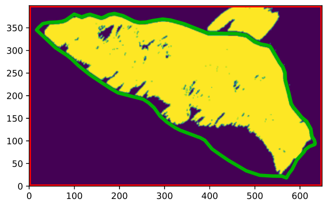
(c) PML: safe
AL: unsafe
RML: unsafe



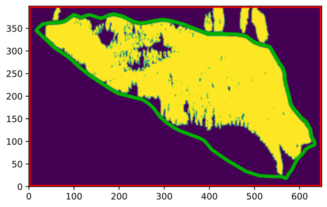
(d) PML: safe
AL: unsafe
RML: unsafe



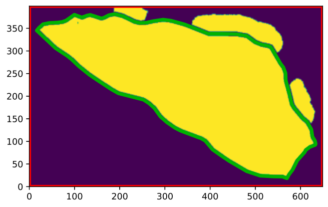
(e) PML: safe
AL: unsafe
RML: unsafe



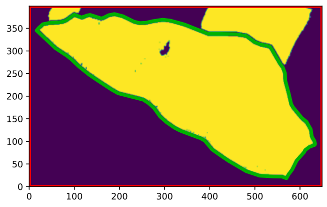
(f) PML: safe
AL: unsafe
RML: unsafe



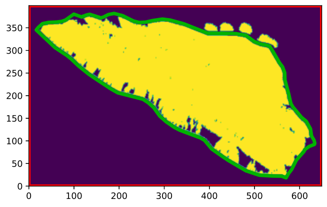
(g) PML: safe
AL: unsafe
RML: unsafe



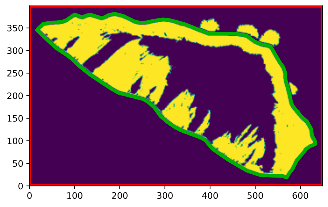
(h) PML: safe
AL: unsafe
RML: unsafe



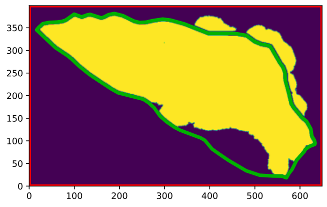
(i) PML: marginal
AL: unsafe
RML: unsafe



(j) PML: safe
AL: unsafe
RML: marginal



(k) PML: marginal
AL: unsafe
RML: unsafe



(l) PML: unsafe
AL: safe
RML: marginal

Figure 5.4. Canonical mismatch cases. PML stands for previous manual label, AL stands for automatic label by applying $\mathbf{u}^1$, and RML stands for revised manual label.

Chapter 6

A Logic Model for Automatically Evaluating Safety of Prescribed Fire

Prescribed fire is more and more accepted as a tool to avoid the extremely hazardous wildfire and balance the fire-dependent ecosystem. Although many researches have been done in the field of wildfire, they might not be applied directly in prescribed fire due to the difference between the uncontrolled wildfire and the planned prescribed fire. Therefore, a comprehensive burn plan to conduct a safe prescribed fire becomes more and more significant. Many fire simulation tools have been developed to simulate the progression of fire with a given wind speed, wind direction, surface moisture and so forth, and an allowable range of these burn conditions such as wind conditions and surface moisture used for the burn plan can be learned from a great amount of simulations (ensembles). Currently, fire domain experts need to take a lot of time and energy to determine the safety of a simulated prescribed fire and to further identify the allowable burn conditions. To reduce the workload of the fire domain experts and improve the efficiency, a logic model is proposed in this chapter to automatically label the safety of a prescribed fire based on the burn area of the prescribed fire simulated by a modern fire simulation tool and the computed and filtered velocity of the fire spread. In other words, in addition to the static information provided by the simulation of the prescribed fire at the last time step, the proposed logic model also takes advantage of the dynamic information of the fire growth rate. Multiple parameters are created in the logic model to evaluate the metrics of fire behavior, and these parameters are optimized by using a subset of the labels manually created by fire domain experts. Afterwards, the logic model with the optimized parameters are used to automatically create the labels for a different subset of simulated prescribed fires, and the automatically created labels are compared to the manually created labels to validate the performance of the logic model. The process of the optimization and the validation are illustrated on 900 simulated prescribed fires in the Yosemite, CA region, created by QUIC-Fire.

6.1 Introduction

Prescribed fire has many beneficial effects on soil, water and vegetation, and is gradually accepted as an effective tool for reducing the hazardous fuels and restoring the ecosystems [15,70]. Various studies related to the effect of the prescribed fire on soil properties are analyzed in [69]. The analysis shows that different burn plans of prescribed fire should be designed specifically for different soil characteristics and vegetation types to achieve the goal of forest management. Compared to wildfire, prescribed fire is a low-intensity fire with less adverse effects on water resource by consuming less organic material, exposing less mineral soil, and causing less mortality of vegetation [78]. Furthermore, long-term suppression of fire leads to difficult regeneration of some woodlands and disappearance of fire-dependent ecosystems. Due to the fact that wildfire is out of control and can result in severe forest damage, prescribed fire is a good tool to sustain and restore the ecosystem [79].

In addition, fuel accumulates by reason of long-term wildfire suppression, and more severe wildfire can happen because of the fuel accumulation [80]. Therefore, instead of suppressing the wildfire all the time, periodic prescribed fire is a better way to avoid uncontrollable and destructive wildfire. The main difference between the wildfire and the prescribed fire is the fire intensity. Many benefits can be brought by a controlled prescribed fire with proper intensity while the escaped prescribed fire can become a severe wildfire and cause great damages. As a consequence, it is important to design a comprehensive burn plan to conduct a safe prescribed fire that can achieve the objective of management. Many researches have been done to study the effect of the prescribed fire on the reduction or suppression of the potential wildfire [81–83]. However, most of them are relied on the empirical or simplified fire model.

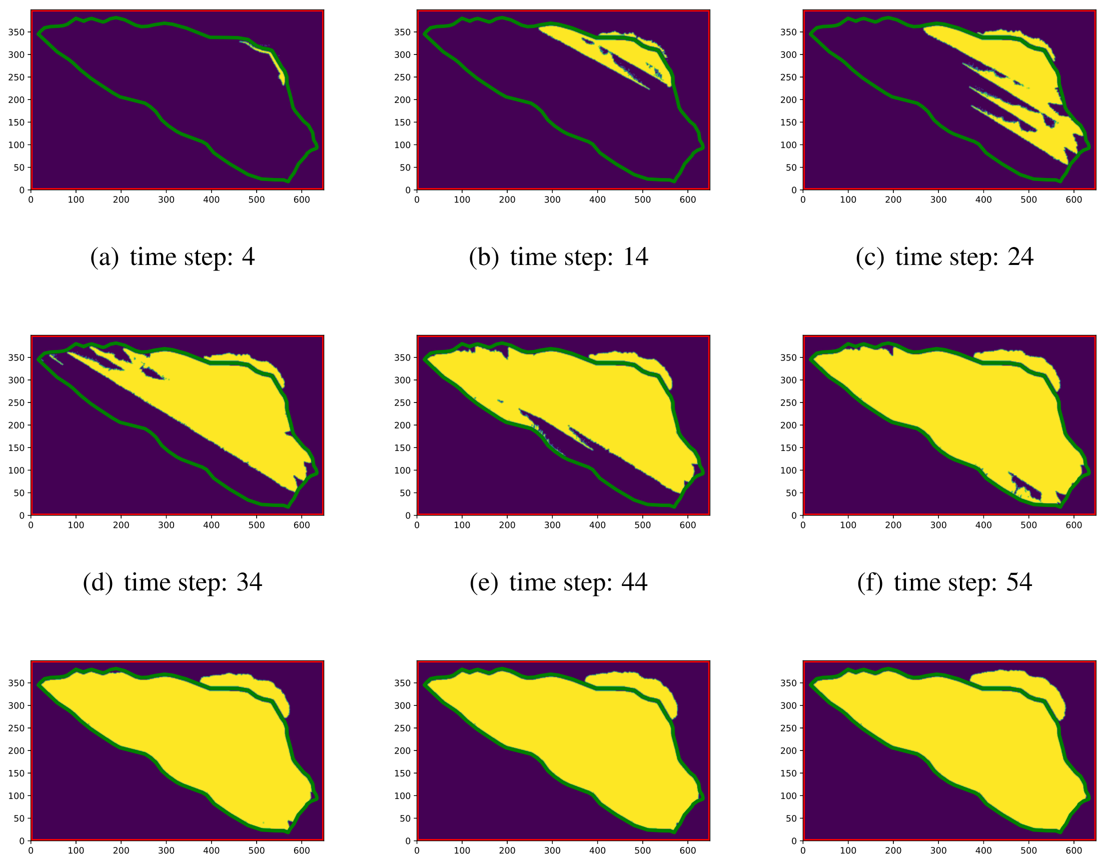
With the advancement of computing power, more and more models have been developed to simulate the progression of fire. A mathematical model is introduced by Rothermel [3]. Subsequently, many empirical or semi-empirical models, BEHAVE [84, 85], FARSITE [8], and Prometheus [86], with a simplified physical process are created. These models use the weather information collected from the weather stations, and take no account of the interaction between the fire and the atmosphere. Fire simulation models are combined with computational fluid dynamics model to capture the fire-atmosphere interactions [9, 11, 87]. In the meantime, most of these models are designed for the wildfire, and wildfire researches might not be proper for the prescribed fire [14]. Considering the difference between the wildfire and the prescribed fire [88], QUIC-Fire is established to predict the progression of the prescribed fire by modeling the response to both ignition patterns and the weather conditions at an appropriate temporal and spatial scale.

Although QUIC-Fire can simulate the spread of the prescribed fire with given burn conditions such as wind conditions, surface moisture, and ignition patterns as a function of time, it cannot be applied directly to provide a burn plan for a prescribed fire due to the complicated non-linear fire progression. Therefore, a large amount of simulations (ensembles) with various burn conditions should be conducted to learn the allowable range of the burn conditions for a particular prescribed fire in a specific area. To label the safety of the simulated prescribed fire and determine the allowable burn conditions, fire domain experts need to view the final burn area of the prescribed fire in each ensemble. This is a tedious process with a heavy workload. To solve this problem, a logic model is proposed in this chapter to automatically labeling the safety of the simulated prescribed fire. Besides the static information of prescribed fire obtained at the final simulation step, the fire growth rate that is calculated based on the images of fuel densities generated by QUIC-Fire at each time step within the simulation is also adopted in the automatic labeling algorithm. Despite that the wind speed and surface moisture can partly reflect the fire growth, the direct application of fire growth rate can make the logic model more general and concise.

For a prescribed fire, a desired burn boundary should be decided in the burn plan for the management objective. Any fire outside the desired burn boundary is denoted as slop-over, and is considered to be the risk of potential wildfire. Multiple metrics including the size, distribution, number and growth rate of the slop-over are extracted from each simulation for the evaluation of fire behavior. Related threshold values are established as the parameters for the logic model to classify these metrics and distinguish the safe, marginal, and unsafe simulated prescribed fire. 900 ensembles of the prescribed fire in the Yosemite, CA region are used for the establishment and validation of the logic model. Due to the fact that the gradient is inaccessible to the logic model, the threshold values in the logic model are optimized via a gradient-free optimization by 48 out of 900 ensembles of prescribed fire with verified manual labels. Afterwards, the logic model is applied to automatically create the labels for the remaining 852 ensembles, and the automatically created labels are compared to the corresponding manually created labels. The results of comparison show that there is 100% match between the automatic labels and the manual labels for the 48 ensembles that are used to optimize the threshold values, and around 80% match for the remaining 852 ensembles. Hence, the proposed logic model is accurate and effective, and can be employed to make the burn plan for the prescribed fire.

6.2 QUIC-Fire Output and Data Processing

The simulated prescribed fire in the Yosemite is ignited at 0s, and the total simulation time is 2100s. The outputs of the QUIC-Fire are generated at 84 time steps with the interval between adjacent time steps being 25s. Two examples of the ensembles are shown in Figure 6.1 and Figure 6.2. Without loss of generality, 9 out of 84 outputs are presented. The yellow area is the burn area, the green line is the desired boundary, and the red line is the allowable boundary. A fire is regarded to escape if any part of the fire crosses the allowable boundary. For Figure 6.1 and Figure 6.2, if a fire (yellow area) finally abuts against the allowable boundary (red rectangle), the fire is regarded to escape. As a result, the first example (Figure 6.1) is a fire that burns within the allowable boundary, and the second one (Figure 6.2) is a fire that escapes outside the allowable boundary.



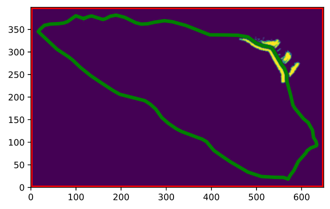
(a) time step: 4 (b) time step: 14 (c) time step: 24

(d) time step: 34 (e) time step: 44 (f) time step: 54

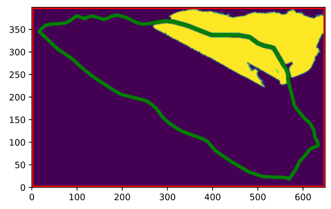
(g) time step: 64 (h) time step: 74 (i) time step: 84

Figure 6.1. A fire that burns within the allowable boundary. The yellow area represents the burn area, the green line represents the desired boundary, and the red line represents the allowable boundary.

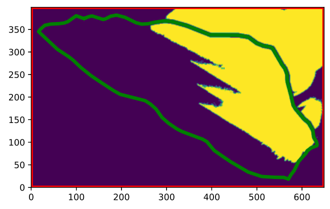
Any fire outside the desired boundary (green line) is a slop-over, and the size, number, distribution, growth rate of the slop-over determine the hazard level of a prescribed fire.



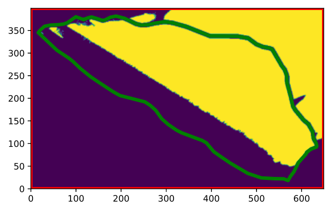
(a) time step: 4



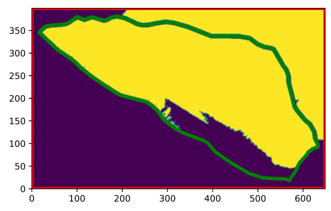
(b) time step: 14



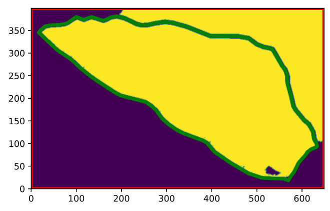
(c) time step: 24



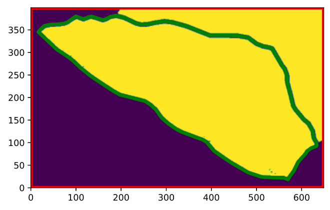
(d) time step: 34



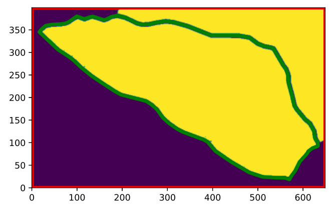
(e) time step: 44



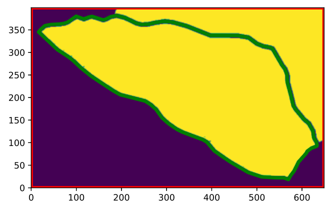
(f) time step: 54



(g) time step: 64



(h) time step: 74



(i) time step: 84

Figure 6.2. A fire that burns outside the allowable boundary. The yellow area represents the burn area, the green line represents the desired boundary, and the red line represents the allowable boundary.

6.2.1 Total Surface Area of Slop-over (A_s)

The size of the slop-over is measured by the total surface area of the slop-over created at the last time step. From Figure 6.1(i) and Figure 6.2(i), it can be observed that the second ensemble (Figure 6.2(i)) is more dangerous in terms of the size of the slop-over. The total surface area is quantitatively calculated by summing the number of pixels of the burn area outside the desired boundary.

6.2.2 Number of Slop-over (k_s)

The number of slop-over is also computed based on the simulation at the last time step. Any isolated burn area outside the desired boundary can be regarded as a single slop-over. Due to the fact that all the pixels of the burn area outside the desired boundary are connected for both Figure 6.1(i) and Figure 6.2(i), the number of slop-over is 1 for both cases.

6.2.3 Maximum Distance Between Slop-overs (l_s)

The distance between slop-overs is calculated by the distance between the center of the smallest rectangle with horizontal and vertical sides that can contain one complete slop-over. The distance between slop-overs is zero if there is only one slop-over. The maximum value of the distance between slop-overs is chosen as the representative value for the distance between slop-overs to display the worst case if there are more than two slop-overs. Similarly, the distance between slop-overs is obtained from the simulation at the last time step.

6.2.4 Maximum Filtered Growth Rate of Slop-over (v_s)

Compared to all other three metrics, the growth rate of the slop-overs is calculated as a function of the time step. In other words, the outputs of the QUIC-Fire at all 84 time steps are required to obtain the growth rate of the slop-over. The growth rate of the slop-over at each time step can be quantified as the difference value between the total surface area of the slop-over at the current time step and the total surface area of the slop-over at the preceding time step. The growth rate of the slop-overs for the two ensembles shown by Figure 6.1 and Figure 6.2 are illustrated by Figure 6.3 and Figure 6.4 respectively. The blue line is the originally computed growth rate. To remove the high frequency components and smooth the growth rate, a second-order Butterworth filter with normalized cutoff frequency 0.1 is applied. The filtered growth rate is represented by the red line. Similar to the distance between slop-overs, the maximum value of the filtered growth rate should be extracted to capture the worst situation of a prescribed fire. From Figure 6.3, it can be observed that the growth rate increases at the beginning while decreases before 30_{th} time step. This phenomenon is due to the effect of the ignition. As shown in Figure 6.1(a), the yellow area inside the desired boundary (green line) is mostly caused by the ignition. The path of the ignition is too close to the desired boundary so that the ignition accelerates the production and the expansion of the slop-over. As a result, for the prescribed fire that burns within the allowable boundary (red line), the maximum value of the filtered growth rate is picked out after the 30_{th} time step to avoid the effect of the ignition. For Figure 6.3, the maximum filtered growth rate is achieved at the last time step.

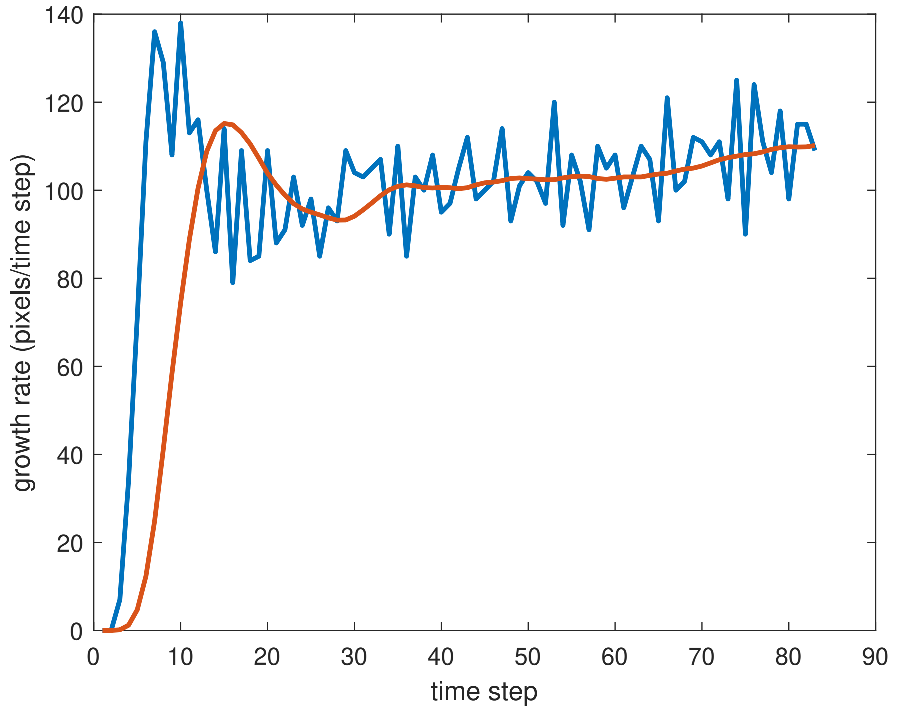


Figure 6.3. Growth rate of the slop-overs in the ensemble shown by Figure 6.1. The blue line is the original growth rate. The red line is the filtered growth rate.

By contrast, for the prescribed fire that escapes outside the allowable boundary (Figure 6.2), it can be observed that the prescribed fire approaches the allowable boundary (red line) in a short time, and almost no change occurring from Figure 6.2(g) to Figure 6.2(i) leads to a small growth rate at the end in Figure 6.4. For these reasons, the effect of the acceleration caused by the ignition is neglected, and the maximum value of the filtered growth rate is sought out from all time steps. For Figure 6.4, the maximum filtered growth rate is achieved at the 14_{th} time step.

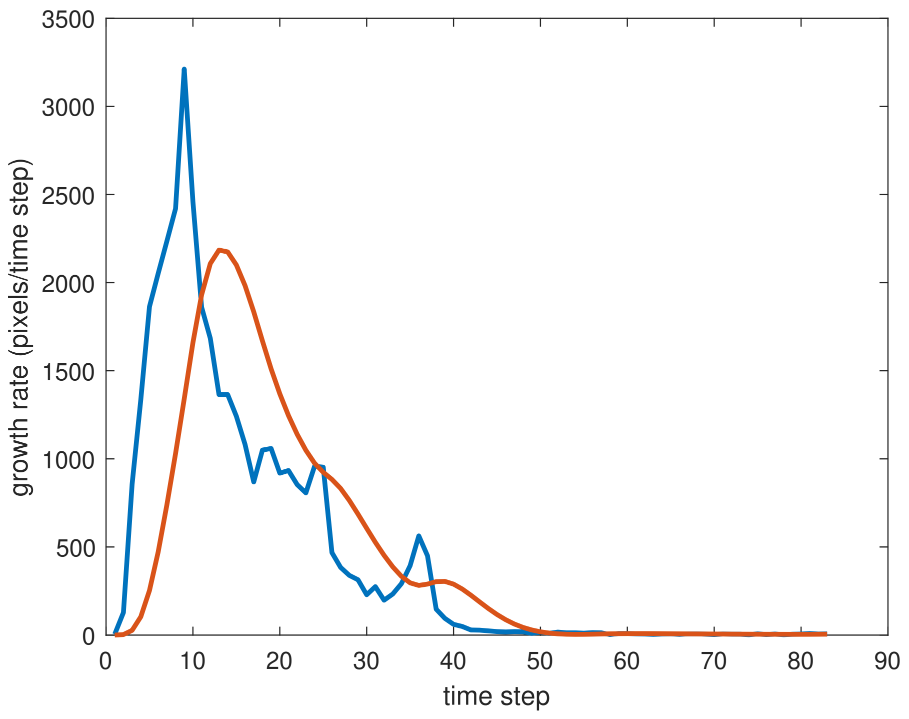


Figure 6.4. Growth rate of the slop-overs in the ensemble shown by Figure 6.2. The blue line is the original growth rate. The red line is the filtered growth rate.

These four metrics, the total surface area of the slop-over (A_s), the number of the slop-over (k_s), the maximum distance between slop-overs (l_s), and the maximum filtered growth rate (v_s), are utilized as the input to the logic model. With different combinations of the four metrics, the fire safety of each ensemble can be decided to be safe, marginal, or unsafe. The parameters required by the logic model to evaluate the four metrics will be introduced in the following section.

6.3 Parameters Definitions

The simulated prescribed fire has the potential ability to escape if one or more metrics is huge. To measure the four metrics, the maximum and marginally allowable values are established for them respectively. If one metric is larger than the corresponding maximum allowable value, the simulated prescribed fire is likely to be unsafe. To make a further judgement, threshold value α is created to evaluate the combined influence of the remaining three metrics. The simulated prescribed fire is identified as an unsafe fire if the risk level in regard to the remaining three metrics is also high enough compared to α. Similarly, a simulated prescribed fire is labeled to be marginal if one metric is larger than its marginal allowable value, and the other three metrics have a great combined influence in comparison to β. For reference, the inputs of the logic model and the maximum and marginally allowable values that are used as the parameters of the logic model are summarized in Table 6.1.

Table 6.1. Nomenclature of inputs and parameters for automatic labeling

Inputs	
A_s	total surface area of slop-over
k_s	number of slop-over
l_s	representative value of distance between slop-overs
v_s	representative value of filtered growth rate of slop-over
Parameters:	
A_{max}	maximum allowed total surface area of slop-overs
A_{mar}	marginally allowed total surface area of slop-over
k_{max}	maximum allowed number of slop-over
k_{mar}	marginally allowed number of slop-over
l_{max}	maximum allowed distance between slop-over
l_{mar}	marginally allowed distance between slop-overs
v_{max}	maximum allowed growth rate of slop-over
v_{mar}	marginally allowed growth rate of slop-over
α	threshold value
β	threshold value

6.4 Logic Model of Automatic Labeling Algorithm

As mentioned before, the prescribed fire has the capability of escaping if one metric is large, and the remaining three metrics should also be checked in order to finally determine the safety of the prescribed fire. With the assumptions as

$$\begin{aligned}
\mathbf{m} &= [A_s, k_s, l_s, v_s], \\
\mathbf{x} &= [A_{max}, k_{max}, l_{max}, v_{max}], \\
\mathbf{r} &= [A_{mar}, k_{mar}, l_{mar}, v_{mar}],
\end{aligned} \tag{6.1}$$

the mathematical expressions of the logic rule for each metric can be established as

$$f_j = \begin{cases} 1, & \text{if } \frac{m_j}{x_j} > 1 \text{ and } \sum\limits_{n=\{1,2,3,4\}\setminus\{j\}} \frac{m_n}{x_n} > \alpha \\ 0, & \text{otherwise} \end{cases}, \tag{6.2}$$

$$g_j = \begin{cases} 1, & \text{if } \frac{m_j}{r_j} > 1 \text{ and } \sum\limits_{n=\{1,2,3,4\}\setminus\{j\}}^{4} \frac{m_n}{r_n} > \beta \\ 0, & \text{otherwise} \end{cases}. \tag{6.3}$$

where $j = 1,2,3,4$ corresponding to each metric respectively. The logic rule means that if the ratio of one metric to its maximum or marginally allowed value is larger than 1, and the summation of the ratios of the remaining three metrics to their maximum or marginally allowed values is larger than the threshold value α or β, the prescribed fire is considered to be unsafe or marginal.

For the logic model of the automatic labeling algorithm, the label of each simulated prescribed fire is assumed to be safe at the beginning. Then, a simulated prescribed fire is regarded to be unsafe if it escapes outside the allowable boundary. For a simulated prescribed fire that burns within the allowable boundary, if $f_j = 1$ for any j, the prescribed fire is labeled to be unsafe. At last, if a prescribed fire is not unsafe, and $g_j = 1$ for any j, the prescribed fire is identified to be marginal. The logic model of the automatic labeling algorithm is summarized in the Algorithm 10.

Algorithm 10. Automatic Labeling Algorithm

```
Inputs: $A_s$, $k_s$, $l_s$, $v_s$
Parameters: $A_{mar}$, $A_{max}$, $k_{mar}$, $k_{max}$, $l_{mar}$, $l_{max}$, $v_{mar}$, $v_{max}$, $\alpha$, $\beta$
Output: Label of the prescribed fire
1: Assume that the prescribed fire is safe.
2: if the prescribed fire escapes outside the allowable boundary then
3:     the prescribed fire is unsafe
4: else if $\sum_{j=1}^{4} f_j > 0$, where $f_j$ is from (6.2) then
5:     the prescribed fire is unsafe
6: else if $\sum_{j=1}^{4} g_j > 0$, where $g_j$ is from (6.3) then
7:     the prescribed fire is marginal
8: end if
```

6.5 Optimization

The performance of the automatic labeling algorithm introduced in Algorithm 10 is principally relied on the value of the parameters summarized in Table 6.1. For different prescribed fire, some typical ensembles with manual labels that are carefully created by fire domain experts can be exploited to optimize the value of the parameters. The formal problem of determining the numerical value of the parameters based on the manually created labels of the fire safety can be stated as the optimization

$$\begin{gathered}\min_{\mathbf{u}} \sum_{i=1}^{10} d_i \frac{u_i}{q_i} - c(\mathbf{u}), \\ \text{subject to: } p_i \le u_i \le q_i, \text{ for } i = 1,2,\ldots,10 \\ u_i \le u_{i+1}, \text{ for } i = 1,3,5,7 \\ c(\mathbf{u}), A_{mar}, A_{max}, k_{mar}, k_{max} \in \mathbb{Z}\end{gathered} \tag{6.4}$$

where $\mathbf{u} = [A_{mar}, A_{max}, k_{mar}, k_{max}, l_{mar}, l_{max}, v_{mar}, v_{max}, \alpha, \beta]$, with u_i represents the i_{th} parameter in $\mathbf{u}$. d_i, p_i, and q_i are the weighting coefficient, lower bound, and upper bound of u_i respectively. The summation of the weighting coefficient is equal to 1 ($\sum_{i=1}^{N} d_i = 1$), and the value of p_i and q_i can be estimated based on the size of the fuel domain and the required management objectives.

$c(\mathbf{u})$ represents the agreement between the labels automatically created by Algorithm 10 and the labels manually created by a fire domain expert.

Considering that the q_i is the upper bound of u_i, and the summation of all weighting coefficients is equal to 1, it can be obtained that $\sum_{i=1}^{10} d_i \frac{u_i}{q_i}$ is less than 1 and any change in the match number $c(\mathbf{u})$ has a greater influence on the optimization. As a result, increasing the goodness of fit between the automatic labels and the manual labels has higher priority than seeking the minimum numerical value of each parameter.

On account of the fact that integer-valued parameters and functions are important components of the logic model, and the gradient of the logic model is inaccessible, a gradient-free optimization is in demand. Among all the gradient-free optimization, genetic algorithm [75] has a good ability to deal with the integer-valued variable. In the meantime, it is convenient to add the essential linear constraints shown in (6.4) during the optimization by using genetic algorithm. Therefore, genetic algorithm is adopted to optimize the numerical values of the parameters.

6.6 Numerical Results

6.6.1 Parameter Optimization

An ensemble of 900 simulated prescribed fires in Yosemite with different burn conditions including the ignition pattern, wind conditions, and the surface moisture is created by QUIC-Fire. The 900 ensembles are divided into two groups as training data set and validation data set. The training data set and validation data set are used for optimizing parameters established in Algorithm 10, and validating the performance of Algorithm 10 in real situation respectively. All these 900 simulated prescribed fires are manually labeled, and 48 out of the 900 ensembles are double checked to ensure the validity of the manual labels. Under this circumstance, those 48 ensembles with validated manual labels are used to optimize the parameters of automatic labeling algorithm, and the automatic labels created by Algorithm 10 with optimized numerical value of the parameters are compared to the manually created labels for the remaining 852 ensembles to evaluate the performance of the automatic labeling.

For $i = 1, 2, \dots, 10$, p_i and q_i in (6.4) are chosen as shown in Table 6.2. Due to the fact that the prescribed fire can be dangerous only when there is at least one slop-over and only finite number of integer values is available for the number of slop-over, p_3 and p_4 are chosen as 1 for the marginally allowed number of slop-over k_{mar} and maximum allowed number of slop-over k_{max} to increase the computational efficiency. Moreover, 5 is chosen for the upper bound of α and β to slightly extend the limitation since all the parameters are minimized in the optimization and lower value than the real one might be obtained numerically.

Table 6.2. The lower bound p_i and upper bound q_i of the parameter u_i.

p_1	p_2	p_3	p_4	p_5	p_6	p_7	p_8	p_9	p_{10}
0	0	1	1	0	0	0	0	0	0
q_1	q_2	q_3	q_4	q_5	q_6	q_7	q_8	q_9	q_{10}
147000	147000	5	5	1000	1000	1000	1000	5	5

With the selected lower bound and upper bound, the optimal values of the parameters obtained by using the genetic algorithm based on the 48 verified manual labels is summarized in Table 6.3 .

Table 6.3. Parameters (**u**) optimized by the 48 verified manual labels.

	A_{mar}	A_{max}	k_{mar}	k_{max}	l_{mar}	l_{max}	v_{mar}	v_{max}	α	β
u	15598	17385	1	2	256.87	323.01	116.91	178.02	1.70	2.39

Algorithm 10 with the parameters summarized in Table 6.3 is applied on all 900 simulated prescribed fires in the Yosemite to automatically create the label for each ensemble.

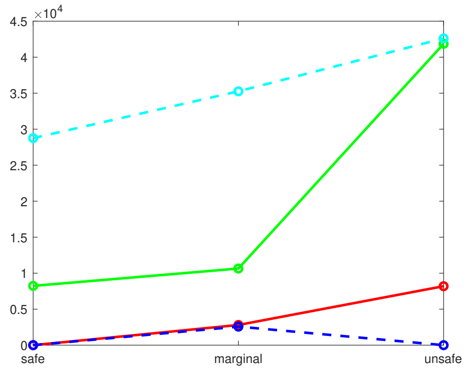
6.6.2 Accuracy of Labeling Matching

100% accuracy has been achieved for the matching between the automatic labels and verified manual labels for the 48 ensembles in the training data set. Hence, the automatic labeling algorithm is capable of automatically creating the labels for the simulated prescribed fires. For the remaining 852 ensembles that are not included in the training data set, another fire domain expert assists in manually labeling the safety of the prescribed fire to share the workload of manual labeling. The accuracy of the matching between the automatic labels and the manual labels for the 852 ensembles with manual labels created by two fire domain experts is 76.29%.

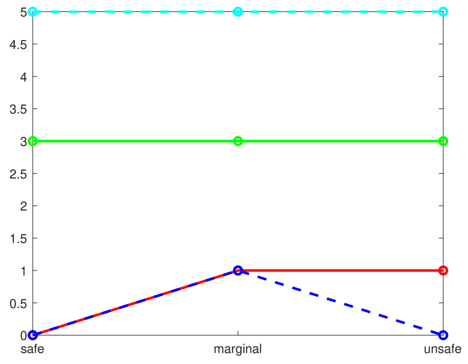
6.6.3 Evaluation of Manual Labeling

Different fire domain experts have different criteria to evaluate the fire safety, and inconsistent manual labels can be created after long hours of work. Due to the fact that the manual labels of the 48 ensembles in the training data set is double checked and more reliable, the manual labels of the remaining 852 ensembles in the validation data set is analyzed. The minimum value and the maximum value of the four metrics are computed for the manually labeled safe, marginal and unsafe prescribed fire based on the 48 ensembles with verified manual labels and the 852 ensembles with mixed manual labels, and the results are illustrated in Figure 6.5.

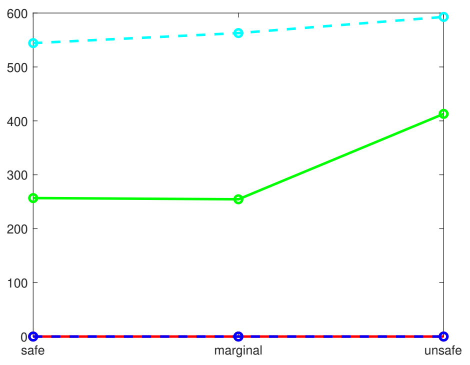
From Figure 6.5(a), it can be observed that the maximum value of A_s of an ensemble from the validation data set (852 ensembles) that is manually labeled as a safe fire is extremely large. Furthermore, the maximum value of A_s of another manually labeled unsafe fire from the validation data set is 0. These are contrary to the reality since the size of the slop-over for a safe fire should be small, and an unsafe fire should have at least one slop-over and the ability to escape. Additionally, the range between the minimum value and the maximum value for the 852 ensembles in the validation data set is much larger than the range for the 48 ensembles in the training data set. It means the training data set is not representative enough, or the manual labels of the validation data set should be reviewed. To figure out the cause of the mismatch between the automatic labels and the manual labels, the fire domain expert who provides the manual labels for the training data set re-labels 12 canonical cases out of the 852 ensembles with mismatched automatic labels and manual labels. 9 of the 12 revised manual labels are the same as the automatic labels, while 3 of the 12 revised manual labels are still different from the automatic labels. Therefore, manual labels can be inconsistent, and automatic labeling algorithm



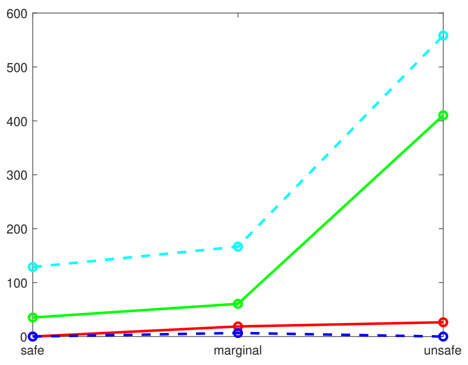
(a) Total surface area of slop-over (A_s)



(b) Number of slop-over (k_s)



(c) Maximum distance between slop-overs (l_s)



(d) Maximum filtered growth rate (v_s)

Figure 6.5. Minimum and maximum value of the four metrics for the manually labeled safe, marginal, and unsafe prescribed fire. Minimum and maximum value for 48 ensembles (red and green), and for 852 ensembles (blue and cyan).

can be applied to correct the manual labels to some extent. In the meantime, more ensembles with verified manual labels should be added to the training data set to improve the optimization of the parameters utilized in the automatic labeling algorithm.

6.6.4 Further Improvements

Although four metrics can all be used for measuring the fire safety, the importance of the four metrics are different. For example, the distance between slop-over is always 0 and provides no information if there is only one slop-over. Hence, weightings can be put on each ratio in (6.2) and (6.3) to label the fire safety more precisely. In addition, more suggestions can be requested from the fire domain experts about the potential value of each parameter in Algorithm 10 to better determine the upper bound and lower bound of each parameter.

6.7 Summary

This chapter proposes a logic model to automatically label the safety of a prescribed fire by evaluating the metrics of the total surface area of the slop-over, the number of the slop-over, the maximum distance between each slop-over, and the maximum filtered velocity of the fire growth. In addition to the static information, the dynamic information of fire growth rate is adopted to establish a more general and comprehensive logic rule. The parameters of the automatic labeling algorithm are optimized by using the genetic algorithm, and 900 simulated prescribed fires in the Yosemite, CA region, with manually created fire safety labels are used for the optimization and validation. 48 of 900 ensembles with verified manual labels are used as the training data and 100% accurate match has been achieved between the automatic labels and manual labels, the remaining 852 of 900 ensembles are used to validate the performance of the logic model, and the match is around 80%. The manual labels of the 852 ensembles used for validation are analyzed, and inconsistency is found. 9 out of 12 revised manual labels of the ensembles from the 852 ensembles are the same as the automatic labels. Therefore, the automatic labeling can be exploited to correct the inconsistent manual labels to some extent. Besides the labels of the fire safety, the logic model of the automatic labeling algorithm can also provide the explanation for the result of labeling based on the fact that which rule in the model is violated.

6.8 Acknowledgement

We thank Matthew Snider and J. Kevin Hiers for manually labeling the ensembles.

Chapter 6, in full, is currently being prepared for submission for publication of the material. Tan, L., de Callafon, R. A., Nguyen, M., & Altıntaş, I. The dissertation author was the primary investigator and author of this material.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This dissertation proposes multiple algorithms to tackle the specific problems in the estimation of wildfire propagation and the safety evaluation of prescribed fires. At first, wind conditions are optimized by comparing the simulated wildfire perimeters and the measured wildfire perimeters using two errors, weighted least-squares error and weighted surface area error. Grid search method is adopted for the optimization, and the optimized wind conditions greatly improve the performance of the wildfire simulations by FARSITE. Then, the iterative minimum distance algorithm is created to automatically generate the closed polygons of wildfire perimeters by ordering the unordered boundary points obtained using image segmentation methods. The ordering of the boundary points is implemented by finding the closest boundary points one by one, and iteration is utilized when there are multiple closest boundary points. Afterwards, iterative trimming method is designed to detect the wildfire perimeter for the thermal infrared image of wildfire with sparse pixels that have high infrared value. Compared to the classical image segmentation methods that can hardly recognize the wildfire perimeter for the TIR image with disconnected burn areas owing to the fact that they are highly relied on the image gradient or image continuity, the iterative trimming method performs well in creating the closed polygon for the main wildfire perimeters. For the prescribed fire, two logic models are introduced to automatically labeling the fire safety of a simulated prescribed fire, multiple metrics including the total surface area of slop-overs, number of slop-overs, and distances between slop-overs are computed to measure the fire risk level. One logic model takes advantage of the wind speed and surface moisture to partially reflect the information of fire growth, while the other one exploits the fire growth rate directly. Both of them can create automatic labels for each simulated prescribed fire. One logic model using the wind speed and surface moisture is carefully designed for various conditions, and the other logic model including the fire growth rate is more general and neater. All these algorithms are illustrated on real data of a wildfire or simulations from fire models.

7.2 Recommendations for Future Work

This dissertation proposes several algorithms to improve the prediction accuracy of the fire progression and detect the fire perimeter. Following these two directions, two more interesting problems need to be solved. The first one is how to describe the fire perimeter with fewer vertices so that the computational effort in data assimilation method can be further reduced, and the second one is how to characterize the uncertainty on each vertex after the closed polygon of fire perimeter is established by the iterative trimming method.

7.2.1 Reduced Vertex Complexity Fire Perimeter

As shown in Figure 2.1, many vertices are closely clustered leading to a problem of the skewed emphasis of the weighted least-squares error. To solve this problem, a reduced vertex complexity closed polygon of the fire perimeter can be a good solution. In addition, vertex complexity reduction can also result in the reduction of computational time in comparing the simulated wildfire perimeter and the measured wildfire perimeter by using the weighted least-squares error. There are three potential directions for obtaining the reduced vertex complexity closed polygon. The first one is to use the Visvalingam–Whyatt algorithm, the second one is to apply a downsampling on the filtered vertices of the closed polygon, and the last one is to find out a simplified piecewise line segments representation of the original fire perimeter. All three approaches are applied on a simulated wildfire perimeter of Maria fire with 413 vertices (Figure 7.1).

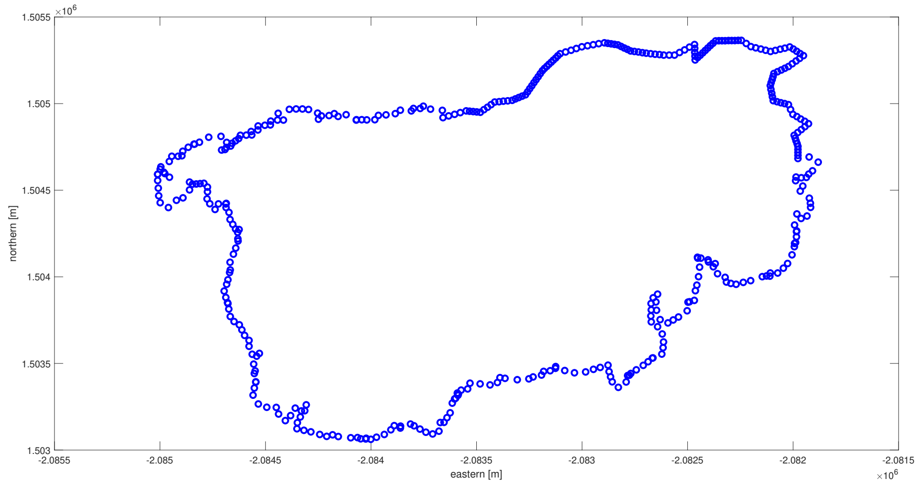


Figure 7.1. One simulated wildfire perimeter of Maria fire. Blue circles represent the 413 vertices of the fire perimeter.

Visvalingam–Whyatt Algorithm

Visvalingam–Whyatt algorithm [89] is an algorithm to represent a curve composed of line segments by a similar curve with fewer vertices. Each vertex of the original curve is assigned a weighting, and these weightings are utilized to remove the less important vertices. With given vertex (x_i, y_i), the weighting of each vertex is calculated as the area of the triangle formed by this vertex and the two adjacent neighboring vertices (x_{i-1}, y_{i-1}) and (x_{i+1}, y_{i+1}) by using the formula:

$$A_i = \frac{1}{2}\left|x_{i-1}y_i + x_iy_{i+1} + x_{i+1}y_{i-1} - x_{i-1}y_{i+1} - x_iy_{i-1} - x_{i+1}y_i\right|.$$

The vertex with the smallest area A_i is removed until the desired number of vertices is reached. One problem of this algorithm is that it will remove all the closely clustered vertices if the desired number of vertices is extremely small because closely clustered vertices normally have relatively smaller A_i. This problem can be solved by removing the vertices iteratively. For the simulated wildfire perimeter with 413 vertices, if the desired number of vertices is chosen to be 50, the result of Visvalingam–Whyatt algorithm without iteration is shown as Figure 7.2.

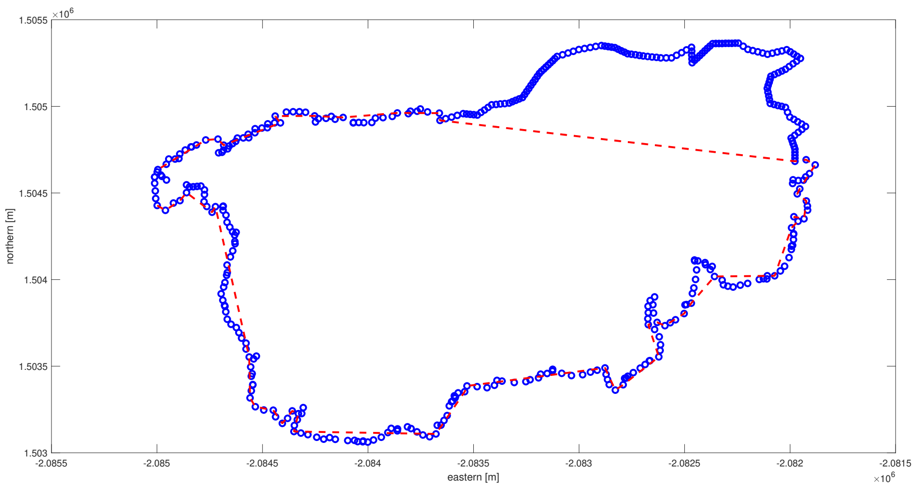


Figure 7.2. Result of Visvalingam–Whyatt algorithm without iteration. Blue circles represent the original 413 vertices of the fire perimeter. Red dashed line represents the reduced vertex complexity fire perimeter.

From Figure 7.2, it can be observed that all the closely clustered vertices in the top right part are removed, so the result of Visvalingam–Whyatt algorithm without iteration is not acceptable. In comparison, if Visvalingam–Whyatt algorithm is used to remove 60 vertices in every iteration, the result is presented as Figure 7.3.

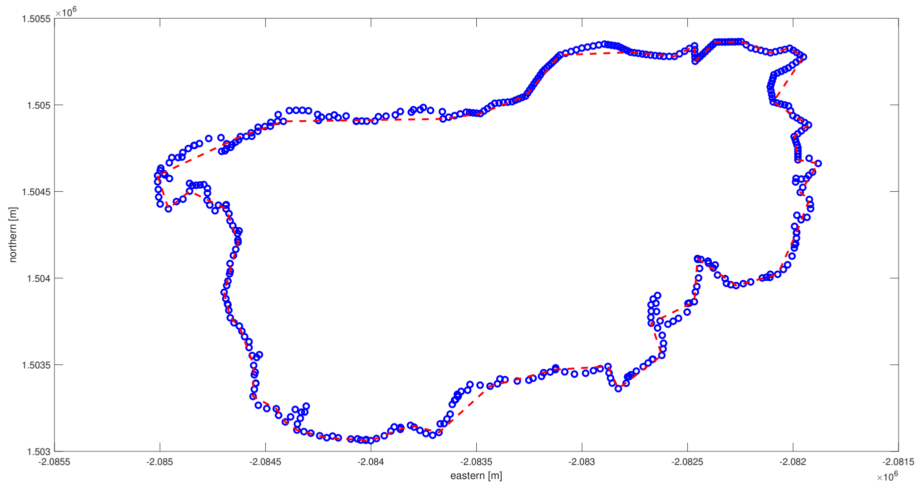


Figure 7.3. Result of iterative Visvalingam–Whyatt algorithm. Blue circles represent the original 413 vertices of the fire perimeter. Red dashed line represents the reduced vertex complexity fire perimeter.

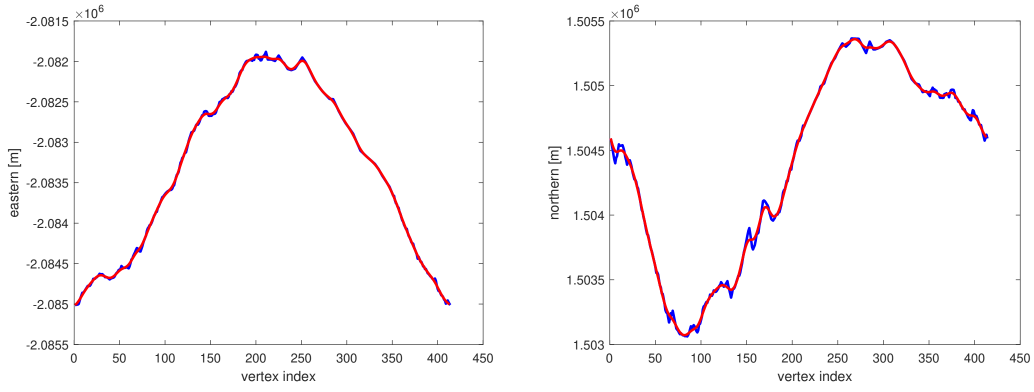
For the iterative Visvalingam–Whyatt algorithm, the reduced vertex complexity closed polygon is decent, but how to optimally choose the number of vertices to be removed in each iteration needs to be figured out.

Filtering and Downsampling Algorithm

Another good direction for obtaining the reduced vertex complexity closed polygon of a fire perimeter is by using the filtering and downsampling method. The first step is to filter the original vertices based on a two-dimensional filtering on x and y coordinates, and the second step is to downsample the filtered vertices. For the filtering step, a low-pass forward and backward filter can be applied, and the results of the filtered x and y coordinates of the vertices in Figure 7.1 are shown in Figure 7.4.

With the filtered coordinates, the filtered fire perimeter can be plotted as Figure 7.5. It can be observed that high-frequency terms are removed, or wiggling parts of the original fire perimeter are smoothed.

Then, based on the filtered vertices, a downsampling can be applied by keeping the first



(a) Filtered *x* coordinates of vertices. (b) Filtered *y* coordinates of vertices.

Figure 7.4. Filtered coordinates of vertices in Figure 7.1. Blue lines represent the original coordinates and red lines represent the filtered coordinates.

sample and every eighth sample after the first one to establish a reduced vertex complexity fire perimeter, and the result is shown in Figure 7.6. It can be observed that the red dashed line almost coincides with the black stars.

After the downsampling, only 52 vertices are needed to approximately represent the original fire perimeter consisting of 413 vertices. Compared to the iterative Visvalingam–Whyatt algorithm, filtering and downsampling algorithm is more robust because the filtering will also smooth the distance between adjacent vertices automatically, and the problem of the closely clustered points is solved automatically. The drawback of the filtering and downsampling method is that the number of remaining vertices is hard to be determined directly since the number of the removed vertices are decided by the combined effect of the original number of vertices and the sample rate of downsampling. In addition to applying a position invariant filtering, a position varying filtering can also be utilized based on the importance of each vertex.

Piecewise Line Segments Algorithm

At last, the least-squares error method can be employed to produce a piecewise line segments representation of the fire perimeter. Three main steps are included in this algorithm. The first step is to keep adding the vertex in sequence and check whether the least-squares error

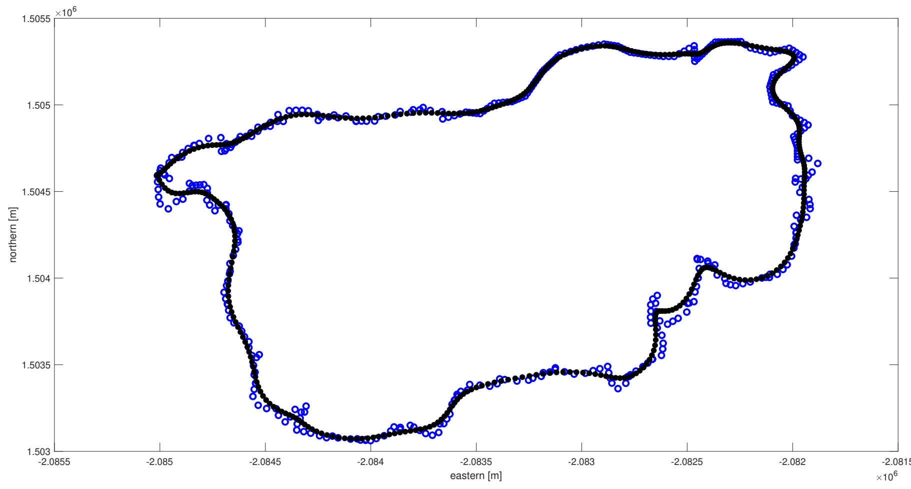


Figure 7.5. Result of filtering. Blue circles represent the original vertices and black stars represent the filtered vertices.

of the created line segment based on the added vertices is larger than a user-defined threshold value. If the error is smaller than the threshold value, then add a new vertex in order. Otherwise, if the error is larger than the threshold value, remove the last added vertex and create the line segments based on the previously added vertices, and then create a new line segment based on the following vertices. After finishing creating the line segments by using all the original vertices, the second step is to combine the adjacent line segments with an extremely close slope. If the slopes of two adjacent line segments are close, then all the vertices used for these two line segments are reused to create a single line segment by using the least-squares error. Finally, connecting the established line segments one by one and calculate the intersection points to describe the closed polygon of the fire perimeter. The result of the line segments representation of Figure 7.1 is shown in Figure 7.7. It can be observed that line segments representation can also capture the main characteristic of the original fire perimeter. The accuracy of the line segments representation can be adjusted by tuning the two threshold values defined for the least-squares error and the tolerance of the slope difference between two adjacent line segments. Still, for the line segment representation, the number of the remaining vertices is decided by the number of line segments and is hardly determined directly by people.

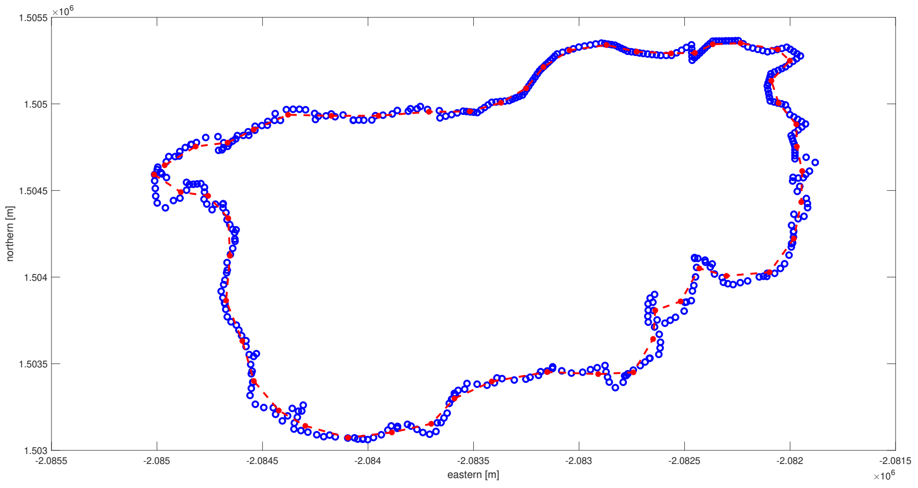


Figure 7.6. Result of downsampling. Blue circles represent the original vertices, red stars represent the downsampled vertices, and red dashed line represents the fire perimeter consisting of downsampled vertices.

Comparison

To evaluate the performance of each method, the results of these three algorithms are compared. The vertex number and the surface area of each reduced vertex complexity fire perimeter are summarized in Table 7.1, and the area of the union of the original fire perimeter and the reduced vertex complexity fire perimeter minus the intersection of the original fire perimeter and the reduced vertex complexity fire perimeter is also calculated for each method.

Table 7.1. Comparison of three reduced vertex complexity fire perimeters.

	Vertex Number	Surface Area	Area of Union minus Intersection
Original Fire Perimeter	413	4509593	0
Visvalingam–Whyatt Algorithm	50	4470309	103028
Filtering and Downsampling Algorithm	52	4434604	138906
Piecewise Line Segments Algorithm	51	4500998	54519

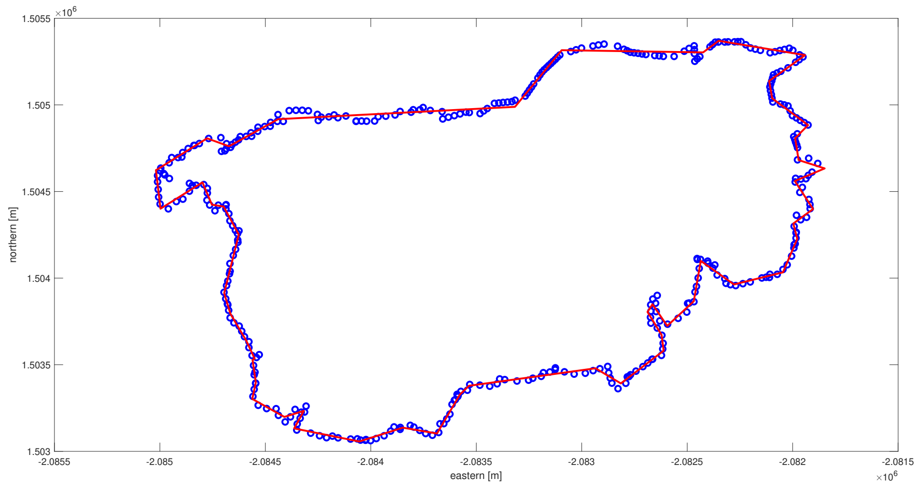


Figure 7.7. Piecewise line segments representation of fire perimeter. Blue circles represent the original vertices and red dashed line represents the piecewise line segments representation of fire perimeter.

7.2.2 Uncertainties on Vertices of Wildfire Perimeter Detected by Iterative Trimming Method

As we introduced before, iterative trimming method can be used to detect the wildfire perimeter for the fire image with sparsely located high-value pixels. In addition to the wildfire perimeter, to describe a two-dimensional uncertainty, a circle can be established for each vertex of the wildfire perimeter based on the average relative burn area surrounding the vertex and the distance from the vertex to the closest neighboring vertices. The average relative burn area can be calculated by the number of the active pixels inside the intersection of the chosen domain around a vertex and the wildfire perimeter divided by the total number of the pixels inside the intersection. Then a high average relative burn area means a small uncertainty on the corresponding vertex because more active pixels provide more information of the burning and vice versa. For the highest average relative burn area, the uncertainty radius is chosen to be 1, and the other uncertainty radius is computed as the value of the highest average relative burn area divided by the corresponding average relative burn area. In the meantime, the closest distance between the vertex and its neighboring vertex can also be exploited to measure the uncertainty. A small closest distance means a small uncertainty on the vertex since vertices are more likely to have large uncertainties if they are sporadically located. With these two values, the uncertainty on each vertex is finally decided by the smaller of the uncertainty radius computed by the average relative burn area and the closest distance from the vertex to its neighboring vertex. The established two dimensional uncertainties on vertices of the wildfire perimeter are illustrated in Figure 7.8. It can be observed that the vertex with a small average relative burn area and a large distance to its neighboring vertex have a relatively larger uncertainty radius.

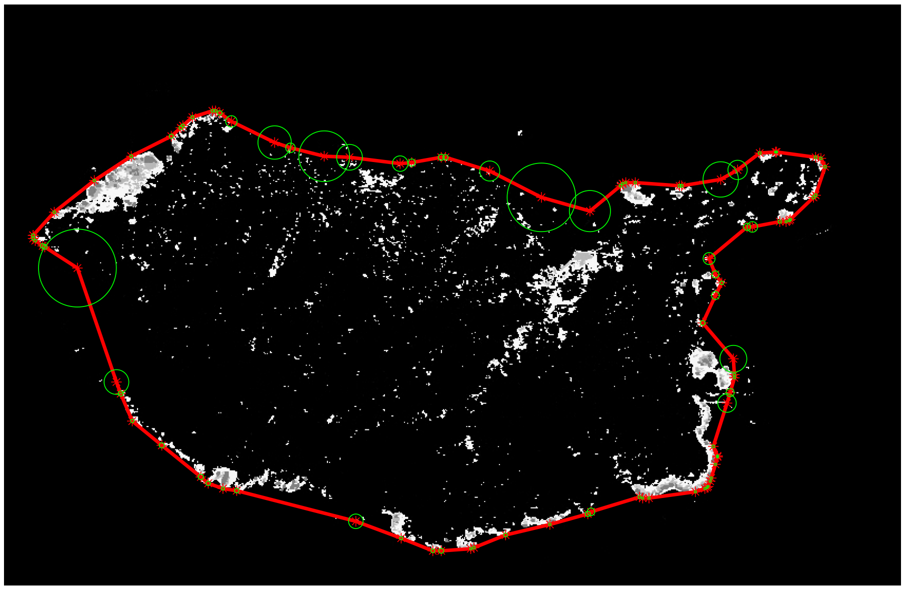


Figure 7.8. Uncertainties on vertices of the wildfire perimeter. White area is the burn area, black area is the unburned area, red line is the wildfire perimeter, and green circles represent the uncertainties on vertices of the wildfire perimeter.

To better visualize the uncertainty regions, an linear interpolation of the uncertainty along the edge of the fire perimeter is carried out based on the discrete uncertainty circles on each vertex. The visualization of the uncertainty regions of the wildfire perimeter is indicated in Figure 7.9.

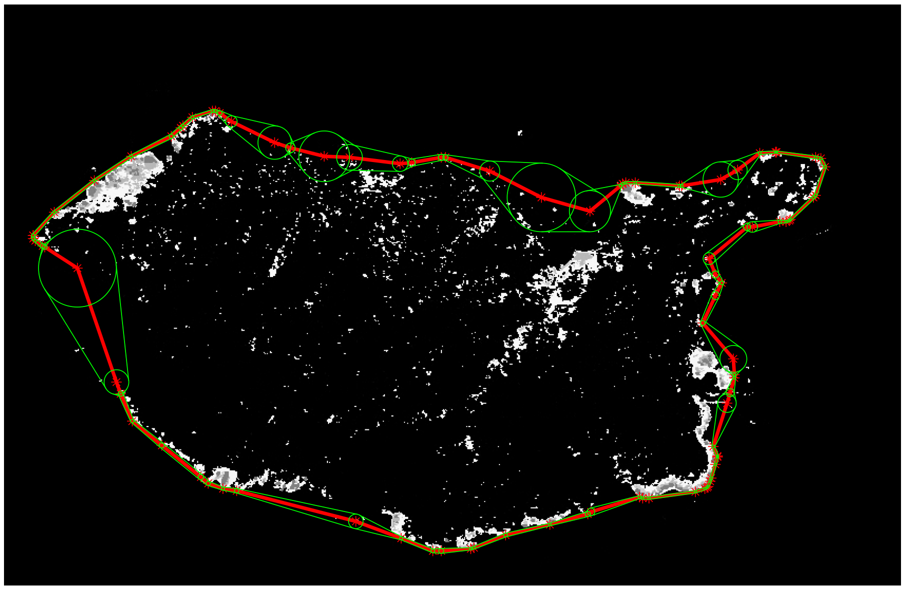


Figure 7.9. Visualization of uncertainty regions (in between the green lines) of the wildfire perimeter. Green circles represent the uncertainties on vertices of the wildfire perimeter.

Bibliography

[1] David MJS Bowman, Jennifer K Balch, Paulo Artaxo, William J Bond, Jean M Carlson, Mark A Cochrane, Carla M D'Antonio, Ruth S DeFries, John C Doyle, Sandy P Harrison, et al. Fire in the earth system. *science*, 324(5926):481–484, 2009.

[2] Sheikh Mansoor, Iqra Farooq, M Mubashir Kachroo, Alaa El Din Mahmoud, Manal Fawzy, Simona Mariana Popescu, MN Alyemeni, Christian Sonne, Jorg Rinklebe, and Parvaiz Ahmad. Elevation in wildfire frequencies with respect to the climate change. *Journal of Environmental management*, 301:113769, 2022.

[3] Richard C Rothermel. *A mathematical model for predicting fire spread in wildland fuels*, volume 115. Intermountain Forest and Range Experiment Station, Forest Service, United ..., 1972.

[4] Jan Mandel, Mingshi Chen, Leopoldo P Franca, Craig Johns, Anatolii Puhalskii, Janice L Coen, Craig C Douglas, Robert Kremens, Anthony Vodacek, and Wei Zhao. A note on dynamic data driven wildfire modeling. In *International Conference on Computational Science*, pages 725–731. Springer, 2004.

[5] Jan Mandel, Lynn S Bennethum, Mingshi Chen, Janice L Coen, Craig C Douglas, Leopoldo P Franca, Craig J Johns, Minjeong Kim, Andrew V Knyazev, Robert Kremens, et al. Towards a dynamic data driven application system for wildfire simulation. In *International Conference on Computational Science*, pages 632–639. Springer, 2005.

[6] Craig C Douglas, Jonathan D Beezley, Janice Coen, Deng Li, Wei Li, Alan K Mandel, Jan Mandel, Guan Qin, and Anthony Vodacek. Demonstrating the validity of a wildfire dddas. In *International Conference on Computational Science*, pages 522–529. Springer, 2006.

[7] Thayjes Srivas, Tomàs Artés, Raymond A De Callafon, and Ilkay Altintas. Wildfire spread prediction and assimilation for farsite using ensemble kalman filtering. *Procedia Computer Science*, 80:897–908, 2016.

[8] Mark A Finney. *FARSITE, Fire Area Simulator – model development and evaluation*, volume 4. US Department of Agriculture, Forest Service, Rocky Mountain Research Station, 1998.

[9] Rodman Linn, Jon Reisner, Jonah J Colman, and Judith Winterkamp. Studying wildfire behavior using firetec. *International journal of wildland fire*, 11(4):233–246, 2002.

[10] Elsa Pastor, Luis Zárate, Eulalia Planas, and Josep Arnaldos. Mathematical models and calculation systems for the study of wildland fire behaviour. *Progress in Energy and Combustion Science*, 29(2):139–153, 2003.

[11] William Mell, Mary Ann Jenkins, Jim Gould, and Phil Cheney. A physics-based approach to modelling grassland fires. *International Journal of Wildland Fire*, 16(1):1–22, 2007.

[12] Andrew L Sullivan. Wildland surface fire spread modelling, 1990–2007. 3: Simulation and mathematical analogue models. *International Journal of Wildland Fire*, 18(4):387–403, 2009.

[13] Rodman Ray Linn, Scott L Goodrick, Sara Brambilla, Michael J Brown, Richard S Middleton, Joseph J O'Brien, and John Kevin Hiers. QUIC-fire: A fast-running simulation tool for prescribed fire planning. *Environmental Modelling & Software*, 125:104616, 2020.

[14] J Kevin Hiers, Joseph J O'Brien, J Morgan Varner, Bret W Butler, Matthew Dickinson, James Furman, Michael Gallagher, David Godwin, Scott L Goodrick, Sharon M Hood, et al. Prescribed fire science: The case for a refined research agenda. *Fire Ecology*, 16:1–15, 2020.

[15] Marcos Francos and Xavier Úbeda. Prescribed fire management. *Current Opinion in Environmental Science & Health*, 21:100250, 2021.

[16] Ingrid Vigna, Angelo Besana, Elena Comino, and Alessandro Pezzoli. Application of the socio-ecological system framework to forest fire risk management: A systematic literature review. *Sustainability*, 13(4):2121, 2021.

[17] Scott L Goodrick, Gary L Achtemeier, Narasimhan K Larkin, Yongqiang Liu, and Tara M Strand. Modelling smoke transport from wildland fires: a review. *International Journal of Wildland Fire*, 22(1):83–94, 2012.

[18] Christopher J Dunn, Matthew P Thompson, and David E Calkin. A framework for developing safe and effective large-fire response in a new fire management paradigm. *Forest Ecology and Management*, 404:184–196, 2017.

[19] Christopher J Dunn, David E Calkin, and Matthew P Thompson. Towards enhanced risk management: planning, decision making and monitoring of us wildfire response. *International journal of wildland fire*, 26(7):551–556, 2017.

[20] Matthew P Thompson, Christopher J Lauer, David E Calkin, Jon D Rieck, Crystal S Stonesifer, and Michael S Hand. Wildfire response performance measurement: current and future directions. *Fire*, 1(2):21, 2018.

[21] Matthew P Thompson, Yu Wei, David E Calkin, Christopher D O'Connor, Christopher J Dunn, Nathaniel M Anderson, and John S Hogland. Risk management and analytics in wildfire response. *Current Forestry Reports*, 5:226–239, 2019.

[22] NP Cheney, JS Gould, and WR Catchpole. The influence of fuel, weather and fire shape variables on fire-spread in grasslands. *International Journal of Wildland Fire*, 3(1):31–44, 1993.

[23] N. Chiaraviglio, T. Artés, R. Bocca, J. López, A. Gentile, J. S. M. Ayanz, A. Cortés, and T. Margalef. Automatic fire perimeter determination using MODIS hotspots information. In *2016 IEEE 12th International Conference on e-Science (e-Science)*, pages 414–423, 2016.

[24] Wilfrid Schroeder, Patricia Oliva, Louis Giglio, Brad Quayle, Eckehard Lorenz, and Fabiano Morelli. Active fire detection using landsat-8/oli data. *Remote Sensing of Environment*, 185:210 – 220, 2016. Landsat 8 Science Results.

[25] Mario M. Valero, Oriol Rios, Christian Mata, Elsa Pastor, and Eulàlia Planas. An integrated approach for tactical monitoring and data-driven spread forecasting of wildfires. *Fire Safety Journal*, 91:835 – 844, 2017. Fire Safety Science: Proceedings of the 12th International Symposium.

[26] Li Tan, Raymond A de Callafon, Jessica Block, Daniel Crawl, and Ilkay Altıntaş. Improving wildfire simulations by estimation of wildfire wind conditions from fire perimeter measurements. In *International Conference on Computational Science*, pages 231–244. Springer, 2021.

[27] Thayjes Srivas, Raymond A de Callafon, Daniel Crawl, and Ilkay Altintas. Data assimilation of wildfires with fuel adjustment factors in farsite using ensemble kalman filtering. *Procedia Computer Science*, 108:1572–1581, 2017.

[28] Huazhen Fang, Thayjes Srivas, Raymond A de Callafon, and Mulugeta A Haile. Ensemble-based simultaneous input and state estimation for nonlinear dynamic systems with application to wildfire data assimilation. *Control Engineering Practice*, 63:104–115, 2017.

[29] Abhishek Subramanian, Li Tan, Raymond A de Callafon, Daniel Crawl, and Ilkay Altintas. Recursive updates of wildfire perimeters using barrier points and ensemble kalman filtering. In *International Conference on Computational Science*, pages 225–236. Springer, 2020.

[30] Zhongjie Lin, Hugh HT Liu, and Mike Wotton. Kalman filter-based large-scale wildfire monitoring with a system of uavs. *IEEE Transactions on Industrial Electronics*, 66(1):606–615, 2018.

[31] Zhewen Xing, Youmin Zhang, Chun-Yi Su, Yaohong Qu, and Ziquan Yu. Kalman filter-based wind estimation for forest fire monitoring with a quadrotor UAV. In *2019 IEEE Conference on Control Technology and Applications (CCTA)*, pages 783–788. IEEE, 2019.

[32] Miguel G Cruz, Andrew L Sullivan, James S Gould, Richard J Hurley, and Matt P Plucinski. Got to burn to learn: the effect of fuel load on grassland fire behaviour and its management implications. *International journal of wildland fire*, 27(11):727–741, 2018.

[33] Miguel G Cruz, Richard J Hurley, Rachel Bessell, and Andrew L Sullivan. Fire behaviour in wheat crops–effect of fuel structure on rate of fire spread. *International journal of wildland fire*, 29(3):258–271, 2020.

[34] Miguel G Cruz, Andrew L Sullivan, and James S Gould. The effect of fuel bed height in grass fire spread: addressing the findings and recommendations of moinuddin et al. *International Journal of Wildland Fire*, 2018.

[35] Daniel Sunday. Fast polygon area and newell normal computation. *journal of graphics tools*, 7(2):9–13, 2002.

[36] Bart Braden. The surveyor's area formula. *The College Mathematics Journal*, 17(4):326–337, 1986.

[37] Crystal A. Kolden and Peter J. Weisberg. Assessing accuracy of manually-mapped wildfire perimeters in topographically dissected areas. *Fire Ecology*, 3:22–31, 2007.

[38] Liu Chun and Tong Xiaohua. Relationship of uncertainty between polygon segment and line segment for spatial data in gis. *Geo-Spatial Information Science*, 8(3):183–188, 2005.

[39] Michael Gollner, Arnaud Trouve, Ilkay Altintas, Jessica Block, Raymond de Callafon, Craig Clements, Anna Cortes, Evan Ellicott, Jean Baptiste Filippi, Mark Finney, et al. Towards data-driven operational wildfire spread modeling: A report of the nsf-funded wifire workshop. Technical report, 2015.

[40] Li Tan, Raymond A de Callafon, Jessica Block, Daniel Crawl, Tolga Çağlar, and Ilkay Altıntaş. Estimation of wildfire wind conditions via perimeter and surface area optimization. *Journal of Computational Science*, page 101633, 2022.

[41] Gary L Achtemeier. Field validation of a free-agent cellular automata model of fire spread with fire–atmosphere coupling. *International Journal of Wildland Fire*, 22(2):148–156, 2012.

[42] F Manzano-Agugliaro, J Pérez-Aranda, and JL De La Cruz. Methodology to obtain isochrones from large wildfires. *International journal of wildland fire*, 23(3):338–349, 2014.

[43] Matthew B Dickinson, Andrew T Hudak, Thomas Zajkowski, E Louise Loudermilk, Wilfrid Schroeder, Luke Ellison, Robert L Kremens, William Holley, Otto Martinez, Alexander Paxton, et al. Measuring radiant emissions from entire prescribed fires with ground, airborne and satellite sensors–rxcadre 2012. *International Journal of Wildland Fire*, 25(1):48–61, 2015.

[44] Thomas J Zajkowski, Matthew B Dickinson, J Kevin Hiers, William Holley, Brett W Williams, Alexander Paxton, Otto Martinez, and Gregory W Walker. Evaluation and use of remotely piloted aircraft systems for operations and research–rxcadre 2012. *International Journal of Wildland Fire*, 25(1):114–128, 2015.

[45] MM Valero, Oriol Rios, E Pastor, and E Planas. Automated location of active fire perimeters in aerial infrared imaging using unsupervised edge detectors. *International journal of wildland fire*, 27(4):241–256, 2018.

[46] Douglas A Stow, Philip J Riggan, Emanual J Storey, and Lloyd L Coulter. Measuring fire spread rates from repeat pass airborne thermal infrared imagery. *Remote sensing letters*, 5(9):803–812, 2014.

[47] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324–331, 1983.

[48] Nicolás Luis Fernández García, Luis Del-Moral Martínez, Ángel Carmona Poyato, Francisco José Madrid Cuevas, and Rafael Medina Carnicer. Unsupervised generation of polygonal approximations based on the convex hull. *Pattern Recognition Letters*, 135:138–145, 2020.

[49] Geir Evensen. The ensemble kalman filter: Theoretical formulation and practical implementation. *Ocean dynamics*, 53(4):343–367, 2003.

[50] Ambrose E Ononye, Anthony Vodacek, and Eli Saber. Automated extraction of fire line parameters from multispectral infrared images. *Remote Sensing of Environment*, 108(2):179–188, 2007.

[51] Piyush Jain, Sean CP Coogan, Sriram Ganapathi Subramanian, Mark Crowley, Steve Taylor, and Mike D Flannigan. A review of machine learning applications in wildfire science and management. *arXiv preprint arXiv:2003.00646*, 2020.

[52] Tom Toulouse, Lucile Rossi, Antoine Campana, Turgay Celik, and Moulay A Akhloufi. Computer vision for wildfire research: An evolving image dataset for processing and analysis. *Fire Safety Journal*, 92:188–194, 2017.

[53] Boris Delaunay et al. Sur la sphere vide. *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk*, 7(793-800):1–2, 1934.

[54] Yasushi Ito. Delaunay triangulation. *Encyclopedia of Applied and Computational Mathematics; Springer: Berlin/Heidelberg, Germany*, pages 332–334, 2015.

[55] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[56] Lijun Ding and Ardeshir Goshtasby. On the canny edge detector. *Pattern recognition*, 34(3):721–725, 2001.

[57] Paul Bao, Lei Zhang, and Xiaolin Wu. Canny edge detection enhancement by scale multiplication. *IEEE transactions on pattern analysis and machine intelligence*, 27(9):1485–1490, 2005.

[58] Weibin Rong, Zhanjing Li, Wei Zhang, and Lining Sun. An improved canny edge detection algorithm. In *2014 IEEE international conference on mechatronics and automation*, pages 577–582. IEEE, 2014.

[59] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[60] Yuri Boykov and Gareth Funka-Lea. Graph cuts and efficient nd image segmentation. *International journal of computer vision*, 70(2):109–131, 2006.

[61] Faliu Yi and Inkyu Moon. Image segmentation: A survey of graph-cut methods. In *2012 international conference on systems and informatics (ICSAI2012)*, pages 1936–1941. IEEE, 2012.

[62] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[63] David Mumford and Jayant Shah. Boundary detection by minimizing functionals. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 17, pages 137–154. San Francisco, 1985.

[64] Chunming Li, Chenyang Xu, Changfeng Gui, and Martin D Fox. Distance regularized level set evolution and its application to image segmentation. *IEEE transactions on image processing*, 19(12):3243–3254, 2010.

[65] Marc D Abrams. Fire and the development of oak forests. *BioScience*, 42(5):346–353, 1992.

[66] James K Agee. *Fire ecology of Pacific Northwest forests*. Island press, 1996.

[67] Juli G Pausas and Jon E Keeley. A burning story: the role of fire in the history of life. *BioScience*, 59(7):593–601, 2009.

[68] BC Scharenbroch, B Nix, KA Jacobs, and ML Bowles. Two decades of low-severity prescribed fire increases soil nutrient availability in a midwestern, usa oak (quercus) forest. *Geoderma*, 183:80–91, 2012.

[69] M Alcañiz, L Outeiro, M Francos, and X Úbeda. Effects of prescribed fires on soil properties: A review. *Science of the Total Environment*, 613:944–957, 2018.

[70] Kevin C Ryan, Eric E Knapp, and J Morgan Varner. Prescribed fire in north american forests and woodlands: history, current practice, and challenges. *Frontiers in Ecology and the Environment*, 11(s1):e15–e24, 2013.

[71] Rodman R Linn and Philip Cunningham. Numerical simulations of grass fires using a coupled atmosphere–fire model: basic fire behavior and dependence on wind speed. *Journal of Geophysical Research: Atmospheres*, 110(D13), 2005.

[72] Jay D Kerby, Samuel D Fuhlendorf, and David M Engle. Landscape heterogeneity and fire behavior: scale-dependent feedback between fire and grazing processes. *Landscape Ecology*, 22(4):507–516, 2007.

[73] Tirtha Banerjee, Warren Heilman, Scott Goodrick, J Kevin Hiers, and Rod Linn. Effects of canopy midstory management and fuel moisture on wildfire behavior. *Scientific reports*, 10(1):1–14, 2020.

[74] Khalid Moinuddin, Nazmul Khan, and Duncan Sutherland. Numerical study on effect of relative humidity (and fuel moisture) on modes of grassfire propagation. *Fire Safety Journal*, 125:103422, 2021.

[75] Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021.

[76] Li Tan, Raymond A de Callafon, and Ilkay Altıntaş. Characterizing wildfire perimeter polygons from quic-fire. In *International Conference on Computational Science*, pages 611–622. Springer, 2022.

[77] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[78] George E Hahn, T Adam Coates, Roger Earl Latham, and Hamed Majidzadeh. Prescribed fire effects on water quality and freshwater ecosystems in moist-temperate eastern north america. *Natural areas journal*, 39(1):46–57, 2019.

[79] Daniel C Dey and Callie Jo Schweitzer. A review on the dynamics of prescribed fire, tree mortality, and injury in managing oak natural communities to minimize economic loss in north america. *Forests*, 9(8):461, 2018.

[80] Ross D Collins, Richard de Neufville, João Claro, Tiago Oliveira, and Abílio P Pacheco. Forest fire management to avoid unintended consequences: A case study of portugal using system dynamics. *Journal of environmental management*, 130:1–9, 2013.

[81] W Lachlan McCaw. Managing forest fuels using prescribed fire–a perspective from southern australia. *Forest Ecology and Management*, 294:217–224, 2013.

[82] Paulo M Fernandes. Empirical support for the use of prescribed burning as a fuel treatment. *Current Forestry Reports*, 1:118–127, 2015.

[83] David A Davim, Carlos G Rossa, José MC Pereira, and Paulo M Fernandes. Evaluating the effect of prescribed burning on the reduction of wildfire extent in portugal. *Forest Ecology and Management*, 519:120302, 2022.

[84] Patricia L Andrews. *BEHAVE: fire behavior prediction and fuel modeling system: BURN subsystem, Part 1*, volume 194. US Department of Agriculture, Forest Service, Intermountain Research Station, 1986.

[85] Patricia L Andrews. Current status and future needs of the behaveplus fire modeling system. *International Journal of Wildland Fire*, 23(1):21–33, 2013.

[86] Cordy Tymstra, RW Bryce, BM Wotton, SW Taylor, OB Armitage, et al. Development and structure of prometheus: the canadian wildland fire growth simulation model. *Natural Resources Canada, Canadian Forest Service, Northern Forestry Centre, Information Report NOR-X-417.(Edmonton, AB)*, 2010.

[87] Rodman Linn, Judith Winterkamp, Carleton Edminster, Jonah J Colman, and William S Smith. Coupled influences of topography and wind on wildland fire behaviour. *International Journal of Wildland Fire*, 16(2):183–195, 2007.

[88] JJ O'Brien, JK Hiers, JM Varner, CM Hoffman, MB Dickinson, ST Michaletz, EL Loudermilk, and BW Butler. Advances in mechanistic approaches to quantifying biophysical fire effects. *Current Forestry Reports*, 4:161–177, 2018.

[89] Maheswari Visvalingam and James D Whyatt. Line generalisation by repeated elimination of points. *The cartographic journal*, 30(1):46–51, 1993.